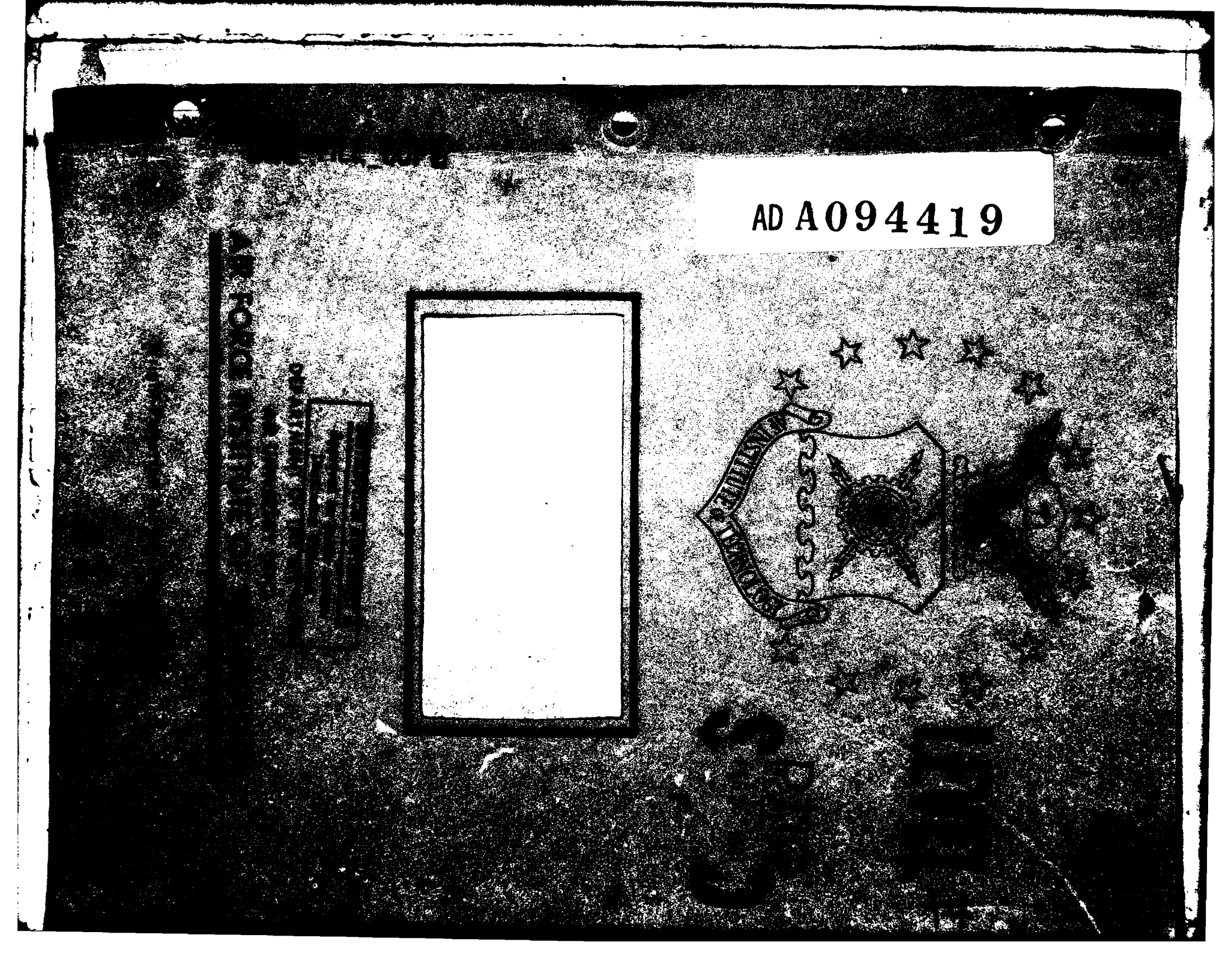
AD-A094 419 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 20/5
LASER WAVE-FRONT ANALYZER SOFTWARE IMPROVEMENT.(U)
DEC 80 R C SUDDUTH
UNCLASSIFIED AFIT/GEO/PH/80-10 NL



AFIT/GEO/PH/80-10

14 JAN 1981

APPROVED FOR PUBLIC RELEASE AFR 190-17.

Fredric C. Lynch

FREDRIC C. LYNCH, Major, USAF
Director of Public Affairs

Air Force Institute of Technology (ATC)
Wright-Patterson AFB, OH 45433

LASER WAVE-FRONT ANALYZER

SOFTWARE IMPROVEMENT

THESIS

AFIT/GEO/PH/80-10 Robert C. Sudduth
2dLt USAF

Approved for public release; distribution unlimited.

(14) AFIT/GEO/PH/80-10

(6) LASER WAVE-FRONT ANALYZER SOFTWARE IMPROVEMENT.

(9) Master's THESIS,

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

by

(10) Robert C. Sudduth B.S.E.E.

2dLt USAF

Graduate Electro-Optics

(12) 215

(11) Dec [redacted] 80

Acknowledgements

Major John D. German of the Air Force Weapons Lab proposed this project. Both he and Jim L. Forgham were a great deal of help to me in defining the limits and requirements of this project.

I would like to thank Dr. Bernard Kaplan for his guidance through the initial stages of development. It was during this time that I learned the basic mathematics required to do the wave-front analysis.

Finally, I want to express my appreciation to my wife, Cindy, for her support and encouragement throughout this project.

Robert C. Sudduth

Contents

List of Figures

List of Tables

Abstract

A software package was written which will analyze annular shaped laser wave-fronts. The polynomials used to estimate the wave-front are based on the work of J.Y. Wang and D.E. Silva in their paper "Wave-front Interpretation with Zernike Polynomials." This involved generating a set of orthogonal polynomials from the Zernike polynomials by using the Gram-Schmidt orthogonalization process. The coefficients of the polynomials are determined by using the orthogonality of the polynomials, instead of using the common least-squares method. The coefficients of the generated polynomials are converted to Zernike coefficients, and both sets of coefficients are presented to the user.

By using the first moments of the wave-front's position in the collection array, the software is able to define the basic parameters of the wave-front. These parameters are: center, outside radius, and the obscuration ratio of the wave-front. With these parameters, the software computes 6, then 11, then 22 coefficients to show the stability of the coefficients. With well-defined circular and annular wave-fronts, the program was consistantly able to compute the coefficients with an RMS error of less than 0.050 waves.

LASER WAVE-FRONT ANALYZER SOFTWARE IMPROVEMENT

I. Introduction

The Laser Wave-front Analyzer (LWA) at the Air Force Weapons Lab (AFWL) is used to measure the phase and intensity of a High Energy Laser (HEL) beam. The LWA uses the phase and intensity data to calculate the coefficients of Patterson's polynomials (Ref.4). The coefficients are used to determine the various optical aberrations present in the laser; the aberrations include: piston, tilt, defocus, and coma. Once the aberrations are known, they can be used to correct the laser's optical elements to maximize the far-field intensity of the laser.

As stated earlier, the Patterson polynomials are currently being used to analyze the wave-front. This set of polynomials is simple and is not the best set of polynomials available. A better set would be the Zernike polynomials modified for an annular wave-front. These polynomials will be used to determine the coefficients which describe the wave-front. Since the polynomials are defined over an annular wave-front, the coefficients will better represent the optical aberrations.

Problem

This thesis involves the generation of a FORTRAN program which will find the coefficients to both Zernike polynomials and a set of polynomials defined over an annular region. Since most HEL's have annular wave-fronts, the Zernike polynomials must be modified to remain orthogonal over the wave-front (the Zernike polynomials are orthogonal with circular wave-fronts). Orthogonality of the polynomials will refer to the integral over the entire region of the product of two polynomials. When the polynomials are the same, the result is one and zero when they are not the same. The coefficients should be available as rapidly as possible after the collection of the data by the LWA.

Assumptions

Work has already been done to show that the Zernike polynomials are not orthogonal when part of the beam is obscured (annular) (Ref.6); therefore, this study will develop the software required to evaluate wave-fronts. It is assumed that the wave-fronts are annuli whose obscuration ratio ranges from 0 to almost 1. The obscuration ratio is the ratio of the inside radius to the outside radius. The software will always try to fit the wave-front with the largest possible annulus. The word wave-front will mean the phase of the laser's output at any instant in time. Thus with an aberration-free system or the Zernike coefficients are all

zero, the wave-front will be planar.

It is assumed that the accuracy of the results will not have to exceed the accuracy of a 16-bit computer (four significant digits to the right of the decimal point); however, the software was developed on a 60 bit computer.

Approach

The general procedure for modifying and finding the coefficients to the Zernike polynomials has been presented in Ref.6 . The basic procedure consists of the following steps: (1) remove any invalid data points, (2) find the center of the beam and determine the obscuration ratio, (3) using the Gram-Schmidt orthogonalization method, compute the modified Zernike polynomials, (4) solve for the aberration coefficients, and (5) print the results. Each of these steps will be presented in detail later.

Development

The basic operation of a LWA and the polynomials used to analyze wave-fronts are presented in Chapter II. The mechanical and software operation of the LWA is shown. In Chapter III, the conversion of Zernike polynomials to a set which are orthogonal over an annulus is presented. Chapter IV deals with the development of the software. The validation of the software is presented in Chapter V, and the conclusions and recommendations are presented in Chapter VI.

II. Detailed Analysis

This chapter deals with the basic workings of a Laser Wave-front Analyzer. Before explaining the mathematics required to analyze a wave-front, the basic operation of the LWA will be discussed. The next two sections present an overview of some of the methods used to analyze wave-fronts. Included in the sections is an explanation of the limitations of these methods.

LWA Operation

The LWA consists of two basic parts, a sliding reference interferometer to collect the data and a computer to control and analyze the data. The inteferometer is designed to work at two wavelength bands: 9.1μm to 10.7μm and 3.2μm to 3.8μm (Ref.7:58). The LWA is capable of collecting the phase gradient and intensity of the beam at each point of a 32 by 32 array 100 times a second. The data is collected by two cryogenically cooled Hg:Cd:Te detectors, one for the X axis data and one for Y axis data. The phase map is assembled later under software control.

The actual phase measurement is done by collecting the difference in phase between a reference beam and the input beam. The input beam is passed through the optical system shown in Figure 1, and is focused on two sets of apertures.

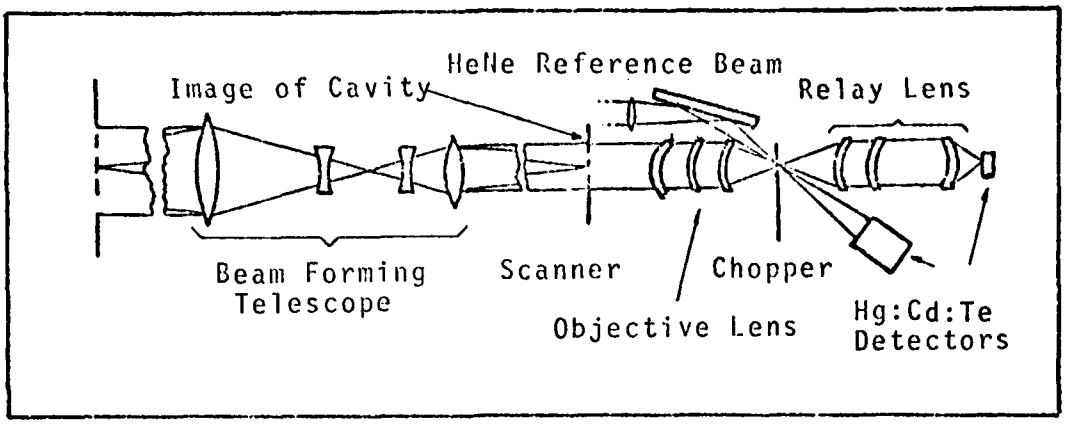


Figure 1. Optical System Layout for IWA

One set is used to find X axis data, and the other Y axis data, with each set consisting of two small circular apertures. The apertures generate Airy diffraction patterns and their proximity to each other generates interference patterns similar to Young's double slit experiment. The optical system focuses the interference pattern on a chopping wheel, which is used to modulate the two beams. The position of the interference pattern is proportional to the optical path difference of the wave-front at the two apertures. Figure 2 shows examples of parallel and tilted wave-fronts. The chopper in Figure 1 modulates the interference pattern into a sine wave. Since a tilted wave-fronts interference pattern is shifted off-axis, its modulated signal will have a corresponding phase shift. In order to measure the phase shift, a reference beam is passed through the same set of apertures and is focused on its own detector, as shown in Figure 1. Since the modulated

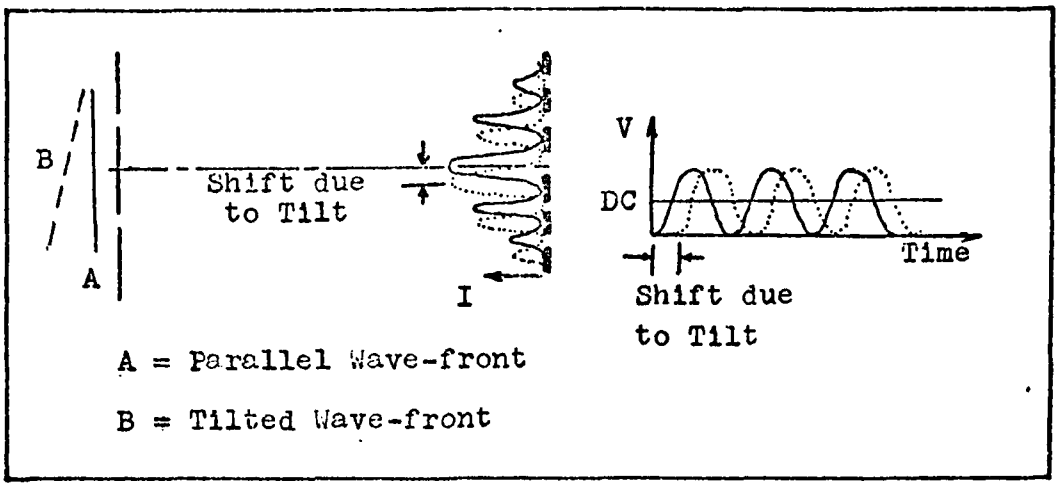


Figure 2. Delta Phase Measurement Procedure

reference beam's signal is constant with respect to time, the phase difference between it and the input beam's signal is used to represent the optical path difference. The LWA is capable of detecting a phase difference as small as $\lambda/200$ (Ref.7:60). The phase difference is called delta phase. The delta phase data is collected at each point in the array by using two orthogonal pair of apertures, resulting in X and Y axis delta phase maps. At the same time the intensity of the wave-front is measured by using the DC value of the modulated signal.

Due to the physical layout of the LWA, the X and Y delta phase is not collected at the same time. The X delta phase is collected starting at row one and ending with row 32. The Y delta phase starts at column 24, decreasing column-wise to one, then to column 32 down to column 25 (Ref.2:6). This collection method is alright as

long as the wave-front does not change shape faster than the sample period of 0.01 seconds; otherwise, the phase measured at one pair of apertures may have changed by the time the other pair of apertures is at the same point.

The delta phase and intensity data is digitized such that the intensity is sent as an eight-bit word and the two delta phases are sent as eleven-bit words, with one bit used to signify valid data. This data is sent as two 16-bit words every 10μsecs to a high speed storage disk to hold the data until it can be processed. The next step in the wave-front analysis is the reconstruction of the wave-front. This is discussed in the next section.

LWA Software Operation

Once the data has been collected by the interferometer, the computer portion of the LWA must first reconstruct the wave-front from the delta phase data, and analyze the wave-front. The analysis includes isometric and isocontour plots, far-field intensity distribution, and polynomial coefficient fitting. This thesis deals only with the coefficient fitting.

Before presenting the current polynomial fitting routine, the method used to reconstruct the wave-front will be presented. The basic technique is to use a least-squares technique with the delta phase. Figure 3 shows a diagram of a basic 3 by 3 wave-front collection matrix. Each W_i point on the array corresponds to a specific value of the

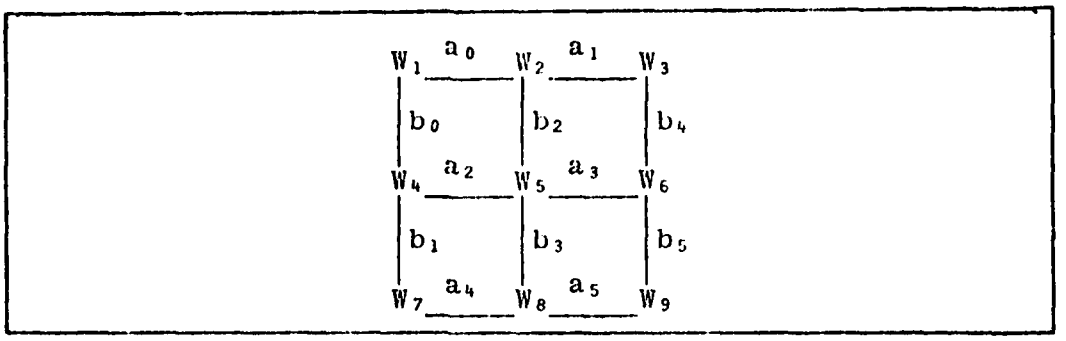


Figure 3. Simple Wave-front Collection Array

wave-front. The a_i's and b_i's correspond to the measured delta phase in the X and Y direction respectively. At each point in the matrix the sum of the square of the differences is computed. For example at the point W_5 the difference equation is

$$\sigma_5 = (W_6-W_5-a_3)^2 + (W_5-W_4-a_2)^2 + (W_8-W_5-b_3)^2 + (W_5-W_2-b_2)^2,$$

Each of the σ_i equations are differentiated with respect to W_i . All of the equations are presented in Appendix A. For this example, the result is nine equations which can be solved simultaneously. As a reference, one of the points is set to zero. The resulting matrix equation is

$$\underline{A}\ \underline{W} = \underline{B} \qquad (2.2.1)$$

where $\underline{W}$ is a column matrix on the actual phase values of the wave-front, $\underline{B}$ is formed from the measured delta phase, and $\underline{A}$ is formed from the difference equations. In this

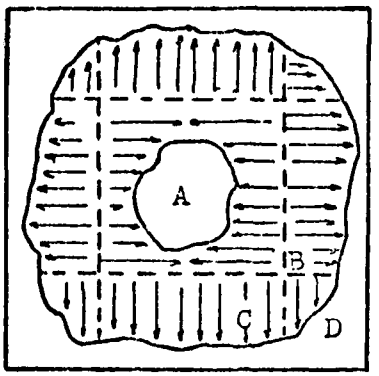
and all other matrix equations, $\underline{A}$ represents a n by n matrix, $\underline{A}$ represents a n by 1 column matrix, and A represents a scalar. It has been shown by others (Ref.5) that $\underline{A}$ in Equation (2.2.1) is Hermitian, irreducibly diagonally dominant, and all diagonal entries are positive real numbers.

The solution is slightly more involved than inverting $\underline{A}$ and multiplying times $\underline{B}$ to get $\underline{W}$. Some of the points in the delta phase may be invalid because of the annular region of the wave-front. The software computes $\underline{B}$ by trying to find dominant rows and columns which circumscribe the annular region. Once the obscured region is "boxed" in, the program works in toward the inside region and outward to the outer radius as shown in Figure 4. This is done to reduce the possible errors introduced by edge effects.

The result of this routine is a reconstructed wave-front which can then be described in terms of various polynomials. The software used now utilizes Patterson's polynomials; more familiar to those in the field of optics are the Zernike polynomials. The next two sections deal with these polynomials.

Patterson Polynomials

The Patterson polynomials are a set of polynomials that were developed at Perkin-Elmer Corporation. They are



A Central Region of Annulus

B Major Axes of "Box"

C Direction of Scan

D Outside Radius of Annulus

Figure 4. Wave-front Construction from Delta Phase Data

basically a simpler set of the Zernike polynomials. Table I shows a comparison of the first 23 Patterson and Zernike polynomials. The most important point about the Patterson polynomials is they are not orthogonal over a unit circle; thus, the coefficients of the polynomials will change as more or fewer of the different polynomials are used in estimating the wave-front. This is the phenomenon the users are faced with at this time. As they increase the number of coefficients being solved for, the previously found coefficients change; therefore, they do not know when the coefficients are the actual coefficients.

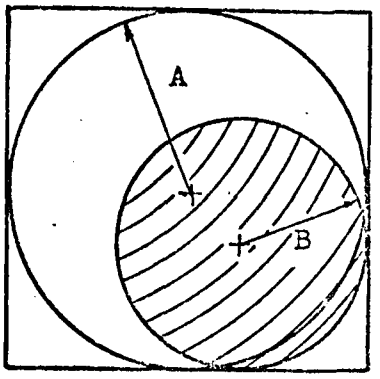
Besides using polynomials which are not orthogonal, the current software assumes that the wave-front will fill the 32x32 collection array, and that it is also centered on the same. Thus an off-center, small beam will introduce much more error than the non-orthogonality of the Patterson polynomials. Figure 5 shows this case. The Patterson polynomials

Table I

Polynomial Comparison

Patterson (Ref 4)		Zernike (Ref 8)	
P(1)	1	Z(1)	1
P(2)	$r\cos\theta$	Z(2)	$2\ r\cos\theta$
P(3)	$r\sin\theta$	Z(3)	$2\ r\sin\theta$
P(4)	r^2	Z(4)	$\sqrt{3}\ (2r^2-1)$
P(5)	$r^2\cos 2\theta$	Z(5)	$\sqrt{6}\ r^2\sin 2\theta$
P(6)	$r^2\sin 2\theta$	Z(6)	$\sqrt{6}\ r^2\cos 2\theta$
P(7)	$r^3\cos\theta$	Z(7)	$\sqrt{8}\ (3r^3-2r)\sin\theta$
P(8)	$r^3\sin\theta$	Z(8)	$\sqrt{8}\ (3r^3-2r)\cos\theta$
P(9)	$r^3\cos 3\theta$	Z(9)	$\sqrt{8}\ r^3\sin 3\theta$
P(10)	$r^3\sin 3\theta$	Z(10)	$\sqrt{8}\ r^3\cos 3\theta$
P(11)	r^4	Z(11)	$\sqrt{5}\ (6r^4-6r^2+1)$
P(12)	$r^4\cos 2\theta$	Z(12)	$\sqrt{10}\ (4r^4-3r^2)\cos 2\theta$
P(13)	$r^4\sin 2\theta$	Z(13)	$\sqrt{10}\ (4r^4-3r^2)\sin 2\theta$
P(14)	$r^4\cos 4\theta$	Z(14)	$\sqrt{10}\ r^4\cos 4\theta$
P(15)	$r^4\sin 4\theta$	Z(15)	$\sqrt{10}\ r^4\sin 4\theta$
P(16)	$r^5\cos\theta$	Z(16)	$\sqrt{12}\ (10r^5-12r^3+3r)\cos\theta$
P(17)	$r^5\sin\theta$	Z(17)	$\sqrt{12}\ (10r^5-12r^3+3r)\sin\theta$
P(18)	$r^5\cos 3\theta$	Z(18)	$\sqrt{12}\ (5r^5-4r^3)\cos 3\theta$
P(19)	$r^5\sin 3\theta$	Z(19)	$\sqrt{12}\ (5r^5-4r^3)\sin 3\theta$
P(20)	$r^5\cos 5\theta$	Z(20)	$\sqrt{12}\ r^5\cos 5\theta$
P(21)	$r^5\sin 5\theta$	Z(21)	$\sqrt{12}\ r^5\sin 5\theta$
P(22)	r^6	Z(22)	$\sqrt{7}\ (20r^6-30r^4+12r^2-1)$

A Radius and Center used in Analysis



B Radius and Center of Actual Wave-front

Figure 5. Possible Error Condition for Input Beam

will give a wave-front which appears to be correct, via the RMS error, but if the same coefficients were used to reconstruct the actual wave-front with the proper coordinates for the center and the true radius, the RMS error would be much higher.

In conclusion, the Patterson polynomials are best suited to give only a rough approximation of the wave-front when it fills the array and is circular. As soon as the wave-front becomes either annular or does not properly fill the collection array, the Patterson polynomials do not even closely represent the aberrations in the wave-front.

Zernike Polynomials

The Zernike polynomials are the classical method used in the field of optics to describe the aberrations present in a wave-front described by a rotationally symmetric optical system. As pointed out in the previous section,

these polynomials are orthogonal over a unit circle. Appendix B (p.105) gives the reader an idea of what the first 22 polynomials look like. As an example, Figure 6 shows the results when the first five coefficients have the value of 0.1 waves, where one wave equals one wave-length of the wave-front.

The Zernike polynomials are well behaved when the wave-front is circular. But when the wave-front becomes an annulus, the Zernike polynomials are no longer orthogonal over the annular region. Thus they are not well suited to the analysis of most HEL beams. Even though the polynomials are not well suited, they are the most familiar to those in optics; therefore, it is desirable to express the wave-front in terms of Zernike coefficients. The next chapter will present a method of converting the coefficients of the newly described polynomials to the coefficients of Zernike polynomials.

Summary

In this chapter the basic operation of the LWA has been presented, along with the way the wave-front is analyzed. The previous discussion has shown that neither the Zernike or the Patterson polynomials are well suited to define the output of most HEL's. The possible sources of error that result from the sampling and analysis have also been presented.

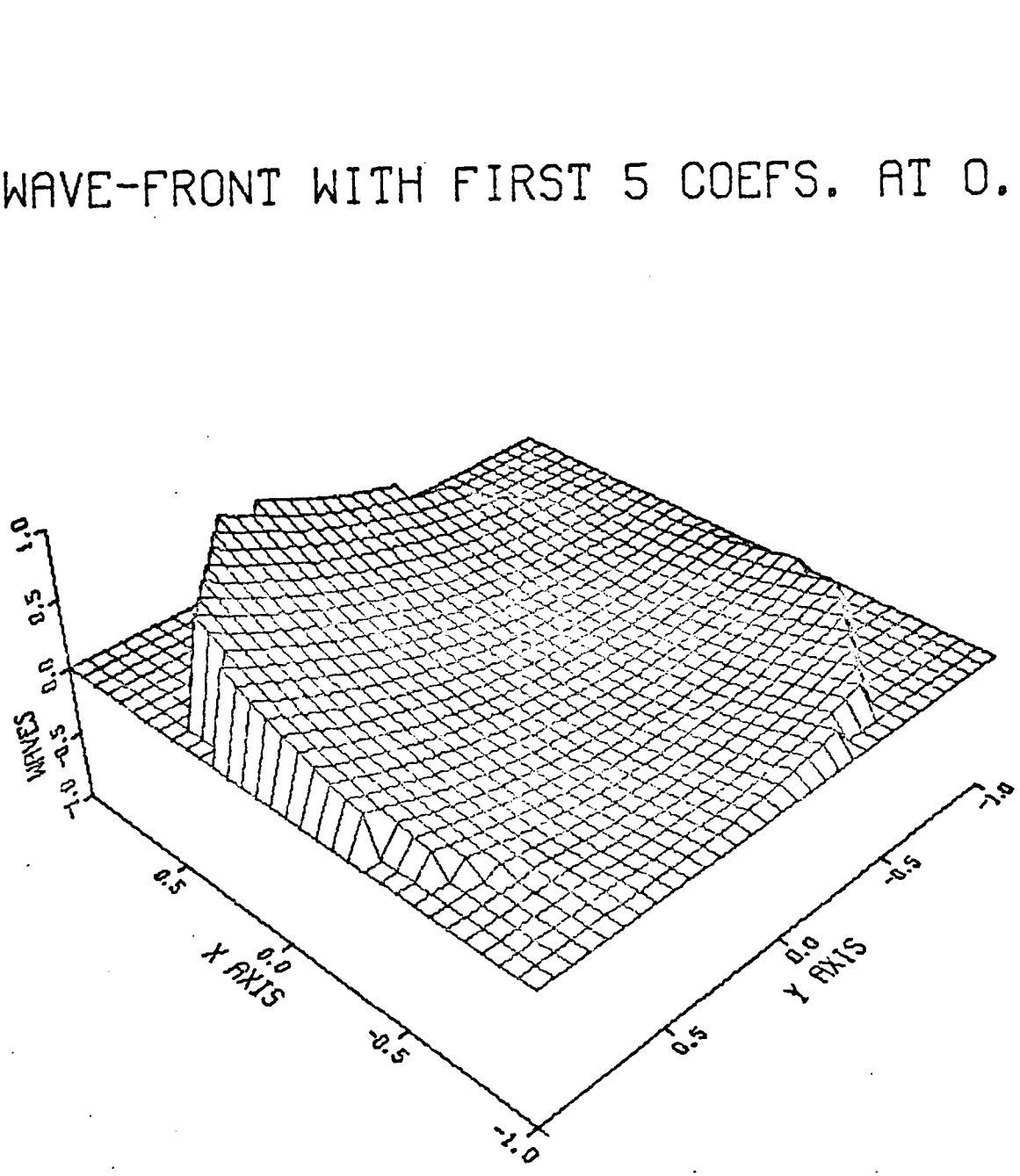


Figure 6. Wave-Front with First 5 Coefficients at 0.1

The next chapter will present a method of generating a set of polynomials which are orthogonal over an annulus. Not only will they be orthogonal over any-sized annulus, but the coefficients to the new set of polynomials can be converted to Zernike coefficients.

III. Theoretical Development

As pointed out in the last chapter, the Zernike polynomials are not orthogonal over a region defined by an annulus. This chapter deals with the theory behind and the development of a set of polynomials which are orthogonal over an annular region with the basic Gram-Schmidt method of orthogonalization being presented first. This will then be applied to the Zernike polynomials, which are defined in more detail, to generate the new set of polynomials orthogonal over an annular domain . Finally, the new polynomials are tested to verify their orthogonality. This chapter is based on the work of Wang and Silva (Ref 8).

Gram-Schmidt Orthogonalization

The Gram-Schmidt orthogonalization process is a method in which a set of linearly independent vectors is combined to form a new set of vectors which are orthogonal. In particular, this process will be applied to the Zernike polynomials. This will be done in the next section; this section deals with the specifics of the Gram-Schmidt orthogonalization process.

The basic theorem for the Gram-Schmidt orthogonalization process is to:

> let the vectors $u_1,\ldots,u_s$ form a set of linearly independent set in a vector space V with inner product

$(\cdot,\cdot)$. Then one can construct an orthogonal set of vectors $x_1,\ldots,x_s$ such that for each i with $1\leq i\leq s$ the set of vectors $x_1,\ldots,x_i$ spans precisely the same subspace as does the set of vectors $u_1,\ldots,u_i$ (Ref.3: 138).

The basic process involved to generate an orthogonal set is to take the first vector u_1 and to normalize it, or

$$v_1 = u_1 \ , \ x_1 = \frac{v_1}{(v_1,v_1)^{\frac{1}{2}}} \tag{3.1.1}$$

where (V_i,V_j) is the inner product of v_i and v_j . To span the same vector space, the second vector u_2 has a multiple of x_1 subtracted from it, or

$$v_2 = u_2 - \alpha_1 x_1 \tag{3.1.2}$$

where α_1 is chosen such that x_1 and v_2 are orthogonal. To find a value for α_1 , the vector x_1 is multiplied times both sides of Equation (3.1.2) and the inner product is then taken to satisfy the orthogonality requirement of x_1 and v_2 giving

$$(x_1,v_2) = (x_1,u_2) - \alpha_1(x_1,x_1) \tag{3.1.3}$$

When two vectors are orthogonal, their inner product is zero. Another important point is that the inner product of a normalized vector with itself is one. Since Equation (3.1.1) is the process used to generate a normalized vector, and x_1 and v_2 must be orthogonal to satisfy the theorem,

Equation (3.1.3) becomes

$$\alpha_1 = (x_1, u_2) \tag{3.1.4}$$

Putting this result into Equation (3.1.2) gives

$$v_2 = u_2 - (x_1, u_2)x_1 \ , x_2 = \frac{v_2}{(v_2, v_2)^{\frac{1}{2}}} \tag{3.1.5}$$

The third vector v_3 follows the same process, thus

$$v_3 = u_3 - \alpha_2 x_2 - \beta_1 x_1 \tag{3.1.6}$$

where α_2 and β_1 are found such that v_3 is orthogonal to x_1 and x_2 .

As an example, just β_1 will be solved for, but the process is the same for α_2 . Thus

$$(x_1, v_3) = (x_1, u_3) - \alpha_2(x_1, x_2) - \beta_1(x_1, x_1) \tag{3.1.7}$$

where again (x_1, u_3) must be zero to satisfy the theorem and (x_1, x_2) is zero because they are two previously defined orthogonal vectors. Equation (3.1.7) becomes

$$\beta_1 = (x_1, u_3) \tag{3.1.8}$$

and in the same manner

$$\alpha_2 = (x_2, u_3) \tag{3.1.9}$$

Using Equations (3.1.8,9) in Equation (3.1.6) gives

$$v_3 = u_3 - (x_2, u_3)x_2 - (x_1, u_3)x_1 \ , \ x_3 = \frac{v_3}{(v_3, v_3)^{\frac{1}{2}}} \tag{3.1.10}$$

This process is continued until all of the desired x vectors are produced (Ref.3:138).

As point out earlier, the above process is the basic means to produce a set of orthogonal vectors. It has been found that this process can be numerically inaccurate (Ref. 3:141), and thus a modified Gram-Schmidt process has been developed. In the original method, x_1 was computed from u_1 and $u_2,\ldots,u_S$ were left alone, with each new x_i vector using only u_i and not affecting the remaining vectors $u_{i+1},\ldots,u_S$. In the modified Gram-Schmidt process, besides computing x_1 from u_1, $(x_1,u_i)x_1$ is subtacted from u_i to produce a new u_i with i ranging from two to S to make u_i orthogonal to x_1 (Ref.3:142). Since u_2^1 is the same as Equation (3.1.5), u_2^1 is normalized to produce x_2. As before this vector is used such that $(x_2,u_i^1)x_2$ is subtracted from u_i^1 to produce u_i^2 with i ranging from three to S to make u_i^2 orthogonal to x_2 as well as x_1 (Ref.3:142). This is continued until x_S has been generated.

The following example shows how the set $\{x_1,x_2,x_3\}$ is generated from $\{u_1,u_2,u_3\}$. As before

$$x_1 = \frac{u_1}{(u_1,u_1)^{\frac{1}{2}}} \tag{3.1.11}$$

but the remaining vectors are modified such that

$$u_2^1 = u_2 - (x_1,u_2)x_1 \tag{3.1.12}$$

$$u_3^1 = u_3 - (x_1, u_3)x_1 \tag{3.1.13}$$

Again from Equation (3.1.5)

$$x_2 = \frac{u_2^1}{(u_2^1, u_2^1)^{\frac{1}{2}}} \tag{3.1.14}$$

which is used to modify u_3^1 giving

$$u_3^2 = u_3^1 - (x_2, u_3^1)x_2 \tag{3.1.15}$$

and

$$x_3 = \frac{u_3^2}{(u_3^2, u_3^2)^{\frac{1}{2}}} \tag{3.1.16}$$

Notice the difference between Equations (3.1.16) and (3.1.10). In Equation (3.1.10) the inner product is taken with an unmodified x_2 and u_3. Thus the modified Gram-Schmidt process gives more accurate results over that of the basic process. It is this modified Gram-Schmidt process which is used to generate the new polynomials. Before presenting the modification of the Zernike polynomials, the basic formulation of the Zernike polynomials is presented.

Zernike Polynomials

The basic characteristics of the Zernike polynomials were discussed earlier, whereas this section deals with their basic construction. Table I showed 22 of the Zernike

Table II

Zernike Radial Polynomials

Radial Degree n	Azimuthal Frequency m						
	0	1	2	3	4	5	6
0	1 Z(1)						
1		r Z(2),Z(3)					
2	$2r^2-1$ Z(4)		r^2 Z(5),Z(6)				
3		$3r^3-2r$ Z(7),Z(8)		r^3 Z(9)Z(10)			
4	$6r^4-6r^2+1$ Z(11)		$4r^4-3r^2$ Z(12)Z(13)		r^4 Z(14)Z(15)		
5		$10r^5-12r^3+3r$ Z(16)Z(17)		$5r^5-4r^3$ Z(18)Z(19)		r^5 Z(20)Z(21)	
6	$20r^6-30r^4+12r^2-1$ Z(22)		$15r^6-20r^4+6r^2$ Z(23)Z(24)		$6r^6-5r^4$ Z(25)Z(26)		r^6 Z(27)Z(28)

polynomials. It can be seen that each polynomial consists of the product of a radial and an angular term. Since the new set of polynomials deals with annular regions, the polynomials will be independent of any angular functions; therefore, it is desirable to separate these two terms.

The means of deriving the radial terms of the Zernike polynomials has already been done by others (Ref.1:sec.9.2). The basic process is described in Appendix C. Table II shows the radial polynomials and all of the corresponding Zernike polynomials. Each radial polynomial was produced by

$$R_n^m(r) = \sum_{s=0}^{(n-m)/2} \frac{(-1)^s (n-s)!\, r^{n-2s}}{s!\left[(n+m)/2 - s\right]!\left[(n-m)/2 - s\right]!} \qquad (3.2.1)$$

where m and n are radial degree and azmuthal frequency respectively. The following conditions must also be met with m and n, such that $m \leq n$ and n minus m is even.

The Zernike polynomials are generated by the following set of equations:

$$\left.\begin{aligned} Z_{\text{even } j} &= \left[2(n+1)\right]^{\frac{1}{2}} R_n^m(r) \cos m\theta \\ Z_{\text{odd } j} &= \left[2(n+1)\right]^{\frac{1}{2}} R_n^m(r) \sin m\theta \end{aligned}\right\} m \neq 0$$
$$\left.\begin{aligned} Z_j &= \left[(n+1)\right]^{\frac{1}{2}} R_n^m(r) \end{aligned}\right\} m = 0 \qquad (3.2.2)$$

where j is the mode-ordering number (Ref.8:1510). In other words, if the second Zernike polynomial is desired, j is set to two. Besides the conditions on m and n mentioned earlier, the following properties of the Zernike polynomials exist:

(1) The polynomials are invariant with respect to rotation about the center of the unit circle.

(2) $$R_n^m(r) = R_n^{-m}(r) \tag{3.2.3}$$

(3) $$\int_0^1 R_n^m(r)R_n^m,(r)rdr = \frac{\delta nn'}{2(n+1)} \tag{3.2.4}$$

(4) $$\int d^2\vec{r}W(\vec{r})Z_j(\vec{r})Z_{j'}(\vec{r}) = \delta_{jj'} \tag{3.2.5}$$

where

$$W(\vec{r}) = \begin{cases} 1/\pi & \text{for } |\vec{r}| \leq 1 \\ 0 & \text{for } |\vec{r}| < 0 \end{cases}$$

and

$$\delta ij = \begin{cases} 1 & i=j \\ 0 & i \neq j \end{cases}$$

The first ten Zernike polynomials have other names which are familiar to those in the field of optics. Table III lists these names with the corresponding Zernike polynomial. With the definition of the Zernike polynomials which are orthogonal over an annular region; this is the

Table III

Aberration and Corresponding Zernike Polynomial

Z(1)	Piston
Z(2)	X Tilt
Z(3)	Y Tilt
Z(4)	Defocus
Z(5)	0^0 Astigmatism
Z(6)	45^0 Astigmatism
Z(7)	Y Coma
Z(8)	X Coma
Z(9)	Y Clover
Z(10)	X Clover

topic of the next section.

Generation of Orthogonal Polynomials

In the previous section, the equations which define the Zernike polynomials were presented, along with the specific properties of the Zernike polynomials. In particular, Equation (3.2.5) deals with the orthogonality of the Zernike polynomials and the region of space in which they are valid. In the case of an annulus, the weighting function $W(\vec{r})$ can no longer have the domain of -1 to 1 but must have a domain of $\beta \leq |\vec{r}| \leq 1$, where β is the obscuration ratio.

Since an annulus is independent of any angular function, the orthogonalization procedure need only deal with the radial portion of the Zernike polynomials. Thus the modified Gram-Schmidt procedure will only be performed on the radial portion of the Zernike polynomial. Again referring to the aspect of angular independence, each new polynomial $N_n^m(r)$ will only depend on those radial polynomials with the same azmuthial frequency m, or $N_n^m(r)$ will depend on $N_m^m(r)$, $N_{m+2}^m(r)$, ..., $N_{n-2}^m(r)$.

Starting with $R_0^0(r)$ the modified Gram-Schmidt procedure yields

$$N_0^0(r) = \frac{R_0^0(r)}{(R_0^0(r),R_0^0(r))^{\frac{1}{2}}} \tag{3.3.1}$$

where

$$(R_0^0(r),R_0^0(r)) = \int_\beta^1 R_0^0(r)R_0^0(r)rdr \tag{3.3.2}$$

which comes from the definition of the inner product of two continuous functions. Since $R_0^0(r)$ has no lower order radial terms $(n\neq 0)$, this is the final step for this new polynomial. This is also true for every new polynomial where m equals n, since one of the conditions on m and n with the Zernike radial polynomials required that n must be greater or equal to m. Thus in general

$$N_n^n(r) = \frac{R_n^n(r)}{(R_n^n(r),R_n^n(r))^{\frac{1}{2}}} \qquad (3.3.3)$$

where again in general

$$(R_n^m(r),R_k^j(r)) = \int_\beta^1 R_n^m(r)R_k^j(r)rdr \qquad (3.3.4)$$

and where m , n , j and k are integers and obey the rules set forth in the previous section. Since $N_i(r)$ is already covered in Equation (3.3.3), the next polynomial is $N_2^0(r)$. In this case the vector space is $R_0^0(r)$ and $R_2^0(r)$. $N_0^0(r)$ is found the same as that in Equation (3.3.1) but $R_2^0(r)$ must be modified to

$$R_2^{0\prime}(r) = R_2^0(r)-(N_0^0(r),R_2^0(r))N_0^0(r) \qquad (3.3.5)$$

and therefore

$$N_2^0(r) = \frac{R_2^{0\prime}(r)}{(R_2^{0\prime}(r),R_2^{0\prime}(r))^{\frac{1}{2}}} \qquad (3.3.6)$$

Again Equation (3.3.3) covers the case of $N_2^2(r)$. Since $N_3^1(r)$ is very similar to $N_2^0(r)$, the next new polynomial to be developed will be $N_4^0(r)$. This polynomial will use as its vector space $R_0^0(r)$, $R_2^0(r)$, and $R_4^0(r)$. The process starts just the same as with $N_2^0(r)$ with Equations (3.3.1,5,6). The next step is to modify $R_4^0(r)$ with $N_0^0(r)$ and then with $N_2^0(r)$, thus

$$R_4^{0\prime}(r) = R_4^0(r) - (N_0^0(r), R_4^0(r))N_0^0(r) \qquad (3.3.7)$$

when $N_0^0(r)$ was computed in Equation (3.3.1) and

$$R_4^{0\prime\prime}(r) = R_4^{0\prime}(r) - (N_2^0(r), R_4^{0\prime}(r))N_2^0(r) \qquad (3.3.8)$$

when $N_2^0(r)$ was computed in Equation (3.3.6). Finally

$$N_4^0(r) = \frac{R_4^{0\prime\prime}(r)}{(R_4^{0\prime\prime}(r), R_4^{0\prime\prime}(r))} \qquad (3.3.9)$$

This process is continued until all of the desired $N_n^m(r)$'s have been found. Obviously if 22 polynomials are to be used in the wave-front analysis, the process of computing all of the integrals would be time-consuming. It is possible to reduce this time factor by realizing that once β is found, all of the inner product integrals can be done at once. The next section deals with the generation of these integration terms.

Inner Product Coefficients

As mentioned in the previous section, the fastest way to compute the new radial polynomials would be to compute all of the inner product integrals at once. This process would have to be done in an orderly fashion since each integral is dependent on the previous integral. For example in Equation (3.3.9), the inner product terms are dependent

on the results of the inner product integrals in Equations (3.3.1,6). It would appear that each integral would have to be computed before doing the next one. Mathematically this is true, but since this system of dependent equations will be put into a program, the problem can be solved in a different way. First a set of variables which will ease the computation of solving for the new radial polynomials will be derived.

In Equation (3.3.1), if the inner product in the denominator is changed to

$$\gamma^0_{00} = (R^0_0(r), R^0_0(r))^{\frac{1}{2}} \tag{3.4.1}$$

and by substituting Equation (3.4.1) in Equation (3.3.1) one gets

$$N^0_0(r) = \frac{R^0_0(r)}{\gamma^0_{00}} \tag{3.4.2}$$

where γ^m_{nn} is the inner product of $N^m_n(r)$ and $R^m_j(r)$ for $j=m,m+2,\ldots,n-2$, and the square root of the inner product of $R^m_n(r)$ and $R^m_j(r)$ when $n=j$. To clarify this, the radial polynomial $N^0_4(r)$ will be recomputed making use of the γ terms.

As before, $N^0_4(r)$ uses the vector space $\{R^0_0(r), R^0_2(r), R^0_4(r)\}$. Thus using Equation (3.4.2) as a starting point, the remaining two vectors must be modified yielding

$$R_2^{0\prime}(r) = R_2^0(r)-(N_0^0(r),R_2^0(r))N_0^0(r) \qquad (3.4.3)$$

and

$$R_4^{0\prime}(r) = R_4^0(r)-(N_0^0(r),R_4^0(r))N_0^0(r) \qquad (3.4.4)$$

The inner product terms of Equations (3.4.3,4) become

$$\gamma_{20}^0 = (N_0^0(r),R_2^0(r)) \qquad (3.4.5)$$

and

$$\gamma_{40}^0 = (N_0^0(r),R_4^0(r)). \qquad (3.4.6)$$

Using Equations (3.4.5,6) in Equations (3.4.3,4) yields

$$R_2^{0\prime}(r) = R_2^0(r)-\gamma_{20}^0 N_0^0(r) \qquad (3.4.7)$$

and

$$R_4^{0\prime}(r) = R_4^0(r)-\gamma_{40}^0 N_0^0(r) \qquad (3.4.8)$$

Continuing with the modified Gram-Schmidt procedure gives

$$N_2^0(r) = \frac{R_2^{0\prime}(r)}{(R_2^{0\prime}(r),R_2^{0\prime}(r))^{\frac{1}{2}}} \qquad (3.4.9)$$

with the inner product term becoming

$$\gamma_{22}^0 = (R_2^{0\prime}(r),R_2^{0\prime}(r))^{\frac{1}{2}} \qquad (3.4.10)$$

With $N_2^0(r)$ computed, $N_4^0(r)$ can be found by

$$R_4^{0\,\prime\prime\prime}(r) = R_4^{0\,\prime\prime}(r)-(N_2^0(r),R_4^{0\,\prime\prime}(r))N_2^0(r) \tag{3.4.11}$$

with

$$\gamma_{42}^0 = (N_2^0(r),R_4^{0\,\prime\prime}(r)) \tag{3.4.12}$$

and

$$N_4^0(r) = \frac{R_4^{0\,\prime\prime\prime}(r)}{(R_4^{0\,\prime\prime\prime}(r),R_4^{0\,\prime\prime\prime}(r))^{\frac{1}{2}}} \tag{3.4.13}$$

The final inner product term is

$$\gamma_{44}^0 = (R_4^0(r),R_4^{0\,\prime\prime\prime}(r))^{\frac{1}{2}} \tag{3.4.14}$$

The γ terms can be generalized with

$$\gamma_{nj}^m = \int_\beta^1 N_{m+j}^m(r)R_n^m(r)rdr \tag{3.4.15}$$

where $j=0,2,4,\ldots,n-2$ and

$$\gamma_{nn}^m = \left[\int_\beta^1 (N_n^{m*}(r))^2 rdr\right]^{\frac{1}{2}} \tag{3.4.16}$$

where

$$N_n^{m*}(r) = R_n^m(r)-\gamma_{nn-2}^m N_{n-2}^m(r)-\ldots-\gamma_{nm}^m N_m^m(r) \tag{3.4.17}$$

From Equations (3.4.15,16), the entire set of γ's can be computed.

It appears that all the γ terms do is simplify the equations, but if Equation (3.4.13) is expanded in terms of $R_n^m(r)$'s from Equations (3.4.1-12), the resulting equation is

$$N_4^0(r) = \frac{1}{\gamma_{44}^0}\left[R_4^0(r) - \frac{\gamma_{40}^0}{\gamma_{00}^0}R_0^0(r) - \frac{\gamma_{42}^0}{\gamma_{22}^0}\left(R_2^0(r) - \frac{\gamma_{20}^0}{\gamma_{00}^0}R_0^0(r)\right)\right] \quad (3.4.18)$$

Solving Equation (3.4.18) for $R_4^0(r)$ in terms of γ's and $N_n^m(r)$'s gives

$$R_4^0(r) = \gamma_{44}^0 N_4^0(r) + \gamma_{42}^0 N_2^0(r) + \gamma_{40}^0 N_0^0(r) \quad (3.4.19)$$

Therefore each Zernike radial polynomial can be expressed in terms of the new radial polynomials and vice versa. In general Equation (3.4.19) becomes

$$R_n^m(r) = \gamma_{nm}^m N_m^m(r) + \gamma_{nm+2}^m(r) + \ldots + \gamma_{nn}^m N_n^m(r) \quad (3.2.20)$$

In a previous section the basic requirements of the Zernike polynomials were expressed. These requirements, being slightly modified, should still be met. Thus

$$N_n^m(r) = N_n^{-m}(r) \quad (3.4.21)$$

$$\int_\beta^1 N_n^m(r) N_{n'}^m(r) r dr = \frac{\delta nn'}{2(n+1)} \quad (3.4.22)$$

$$\int d^2\vec{r} W(\vec{r}) NZ_j(\vec{r}) NZ_{j'}(\vec{r}) = \delta_{jj'} \qquad (3.4.23)$$

where

$$W(\vec{r}) = \begin{cases} 1/\pi & \beta \leq |\vec{r}| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

and $NZ_j(\vec{r})$ is the newly defined Zernike polynomial, using the $N_n^m(r)$'s as the radial components. Equation (3.4.21) is valid since $N_n^m(r)$ is made up of $R_n^m(r)$'s which meet this requirement as shown in Equation (3.2.3). The next section deals with the meeting of the remaining requirements.

Orthogonality Tests

As pointed out in the previous section, it is desirable for the new radial polynomials $N_n^m(r)$ to have the same qualities as the Zernike polynomials $R_n^m(r)$. In particular the orthogonality of the radial polynomials are shown in Equations (3.2.3) and (3.4.22) for the Zernike and the new radial polynomials respectively.

As a test of the new radial terms, the first test will be with the polynomials where $m=n$, thus Equation (3.4.22) becomes

$$\int_\beta^1 N_n^n(r) N_{n'}^{n'}(r) r dr \stackrel{?}{=} \frac{\delta_{nn'}}{2(n+1)} \qquad (3.5.1)$$

But from Equation (3.4.20) the $N_n^n(r)$'s can be expressed as

$$N_n^n(r) = \frac{R_n^n(r)}{\gamma_{nn}^n} \tag{3.5.2}$$

where from Table II

$$R_n^n(r) = r^n \tag{3.5.3}$$

Thus Equation (3.5.1) becomes

$$\int_\beta^1 \left(\frac{1}{\gamma_{nn}^n}\right)^2 r^{2n} r dr \stackrel{?}{=} \frac{1}{2(n+1)} \tag{3.5.4}$$

but

$$\gamma_{nn}^n = \left[\int_\beta^1 (R_n^n(r))^2 r dr\right]^{\frac{1}{2}} \tag{3.5.5}$$

from Equations (3.4.16,17). Using Equations (3.5.3) and (3.5.5) and integrating yields

$$\gamma_{nn}^n = \left[\frac{1-\beta^2(n+1)}{2(n+1)}\right]^{\frac{1}{2}} \tag{3.5.6}$$

Putting Equation (3.5.6) into Equation (3.5.4) and integrating gives a result of one and not $1/2(n+1)$ as desired. Since the inner product coefficients are constants once β is found, Equation (3.4.16) can be modified such that

$$\gamma_{nn}^{m} = \left[2(n+1)\int_{\beta}^{1}(N_n^{m*}(r))^2 r dr\right]^{\frac{1}{2}} \qquad (3.5.7)$$

only when $m=n$. It will be shown that this is also true for all values of m . To verify it for all cases of m , Equation (3.5.1) becomes

$$\int_{\beta}^{1} N_n^m(r)N_{n'}^m(r)rdr = \frac{\delta nn'}{2(n+1)} \qquad (3.5.8)$$

where $N_n^m(r)$ can be changed to the Zernike radial polynomials from Equation (3.4.19). This results in Equation (3.5.8) becoming

$$\left(\frac{1}{\gamma_{nn}^m}\right)^2 \int_{\beta}^{1}\left[R_n^m(r)-\gamma_{nm}^m N_n^m(r)-\gamma_{nm+2}^m N_{m+2}^m(r)-\ldots-\gamma_{nn-2}^m N_{n-2}^m(r)\right]^2 rdr$$

$$\stackrel{?}{=} \frac{1}{2(n+1)} \qquad (3.5.9)$$

Using Equation (3.5.7) as an initial guess for γ_{nn}^m , Equation (3.5.9) becomes

$$\frac{\int_{\beta}^{1}\left[R_n^m(r)-\gamma_{nm}^m N_n^m(r)-\gamma_{nm+2}^m N_{m+2}^m(r)-\ldots-\gamma_{nn-2}^m N_{n-2}^m(r)\right]^2 rdr}{2(n+1)\int_{\beta}^{1}(N_n^{m*}(r))^2 rdr}$$

$$\stackrel{?}{=} \frac{1}{2(n+1)} \qquad (3.5.10)$$

Using Equation (3.4.17), the two integrals cancel, giving the proper results; thus, Equation (3.5.7) is valid for all values of $m \leq n$.

The final test is to see if Equation (3.5.8) still holds true when it is changed to

$$\int_{\beta}^{1} N_n^m(r) N_j^m(r) \stackrel{?}{=} \frac{\delta nj}{2(n+1)} \qquad (3.5.11)$$

where $n \neq j$. Since j can not equal n the Kronecker delta (δnj) will be zero; therefore, Equation (3.5.11) should be zero. This integral is very difficult to verify using the general case, since each new radial polynomial $N_n^m(r)$ and $N_j^m(r)$ would have to be expanded similar to Equation (3.4.19). Since m , n , and j are arbitrary, the series becomes too unwieldy to determine the point at which errors may have been introduced; therefore, several test cases were developed using specific values for m , n , and j . Since the process is still very tedious, only a simple case and the final results will be presented.

As a simple test case

$$\int_{\beta}^{1} N_2^0(r) N_0^0(r) r dr \stackrel{?}{=} 0 \qquad (3.5.12)$$

will be tested. Each of the polynomials can be expanded to

make Equation (3.5.12) become

$$\int_{\beta}^{1}\left[\frac{1}{\gamma^{0}_{22}}(R^{0}_{2}(r)-\frac{\gamma^{0}_{20}}{\gamma^{0}_{00}}R^{0}_{0}(r))\right]\left[\frac{1}{\gamma^{0}_{00}}R^{0}_{0}(r)\right]rdr \stackrel{?}{=} 0 \qquad (3.5.13)$$

From Table II and simplifying, this becomes

$$\int_{\beta}^{1}\left[2r^{2}-(1+\frac{\gamma^{0}_{20}}{\gamma^{0}_{00}})\right]rdr \stackrel{?}{=} 0 \qquad (3.5.14)$$

Using Equations (3.4.15,16) and integrating yields $\frac{1}{4}(\beta^{2}-\beta^{4})$ which does not equal zero. Since the equation used to generate γ^{0}_{00} has already been verified, the only error can come from the generation of γ^{0}_{20}. As a first guess Equation (3.4.15) is changed to

$$\gamma^{m}_{nj} = 2(n+1)\int_{\beta}^{1}N^{m}_{m+j}(r)R^{m}_{n}(r)rdr \qquad (3.5.15)$$

but this gives $\beta^{2}-\beta^{4}$ when used in Equation (3.5.12). The next change made worked, making

$$\gamma^{m}_{nj} = 2(m+1)\int_{\beta}^{1}N^{m}_{m+j}(r)R^{m}_{n}(r)rdr \qquad (3.5.16)$$

This equation was found invalid when the test case using $N^{0}_{4}(r)$ and $N^{0}_{2}(r)$ was used. The final form of the equation is

$$\gamma_{nj}^{m} = 2(m+j+1)\int_{\beta}^{1} N_{m+j}^{m}(r)R_{n}^{m}(r)rdr \qquad (3.5.17)$$

This equation was verified using several test cases, which are too tedious to present. With Equations (3.4.20), (3.5.7), and (3.5.17), it is possible to generate all of the new radial polynomials; thereby, using $N_n^m(r)$ in place of $R_n^m(r)$ in the Zernike polynomials, a new set of orthogonal polynomials can be generated. Figures 7 to 11 show the inner product coefficients with respect to the obscuration ratio. With the set of orthogonal polynomials, an annular wave-front can now be more accurately expressed. The method on analyzing the wave-front will be presented next.

Wave-front Analysis

As presented by others (Ref.1:Sec.9.1,2), wave-fronts can be represented as the summation of the Zernike polynomials times their respective coefficients, or

$$\sum_{j=1}^{\infty} a_j Z_j(\vec{r}) = \phi \qquad (3.6.1)$$

where ϕ is the measured wave-front, and a_j is the coefficient to the Zernike polynomial $(Z_j(\vec{r}))$. Usually in optics one is concerned with the lower order Zernike polynomials; therefore, Equation (3.6.1) becomes,

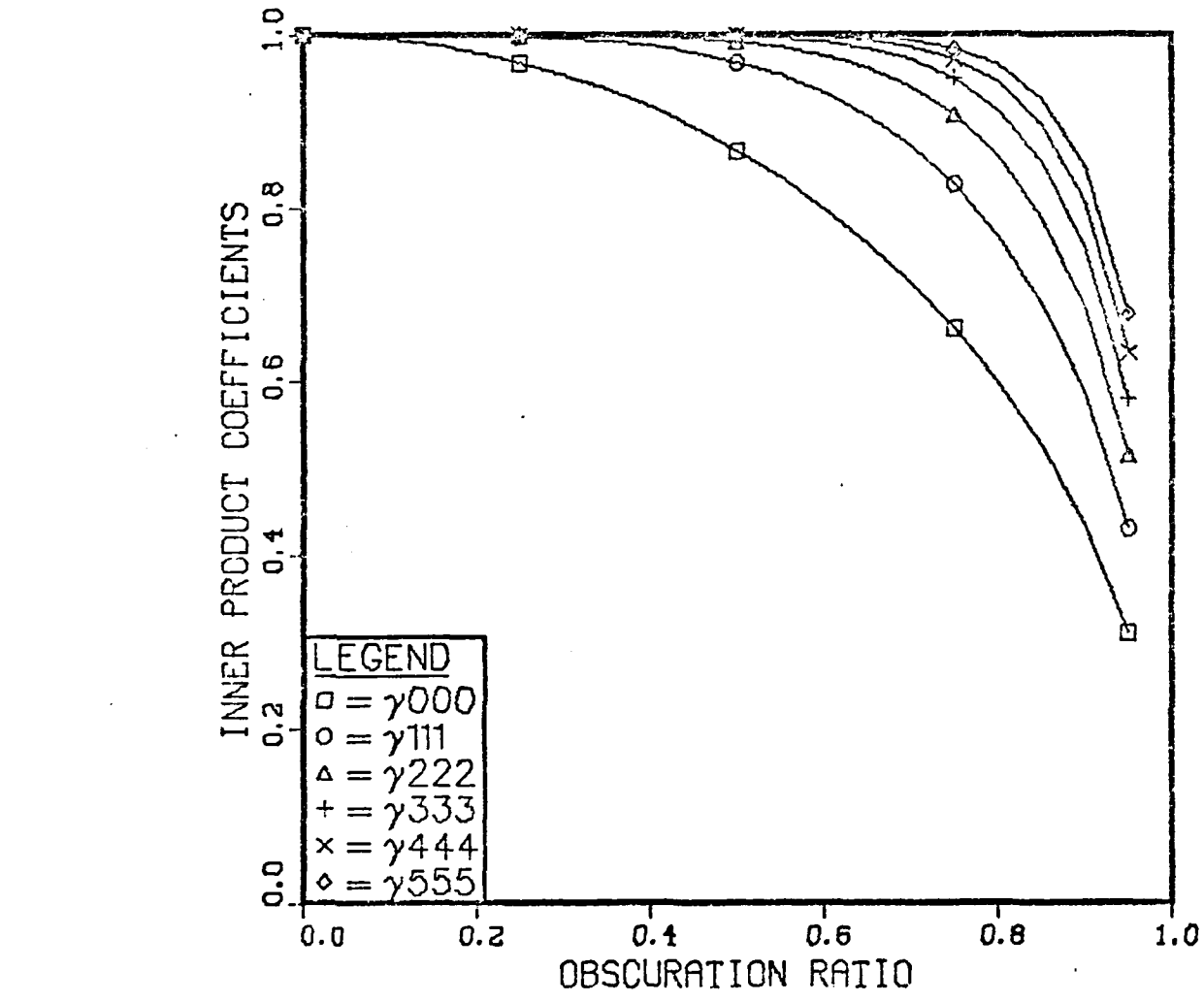


Figure 7. Inner Product Coefficients

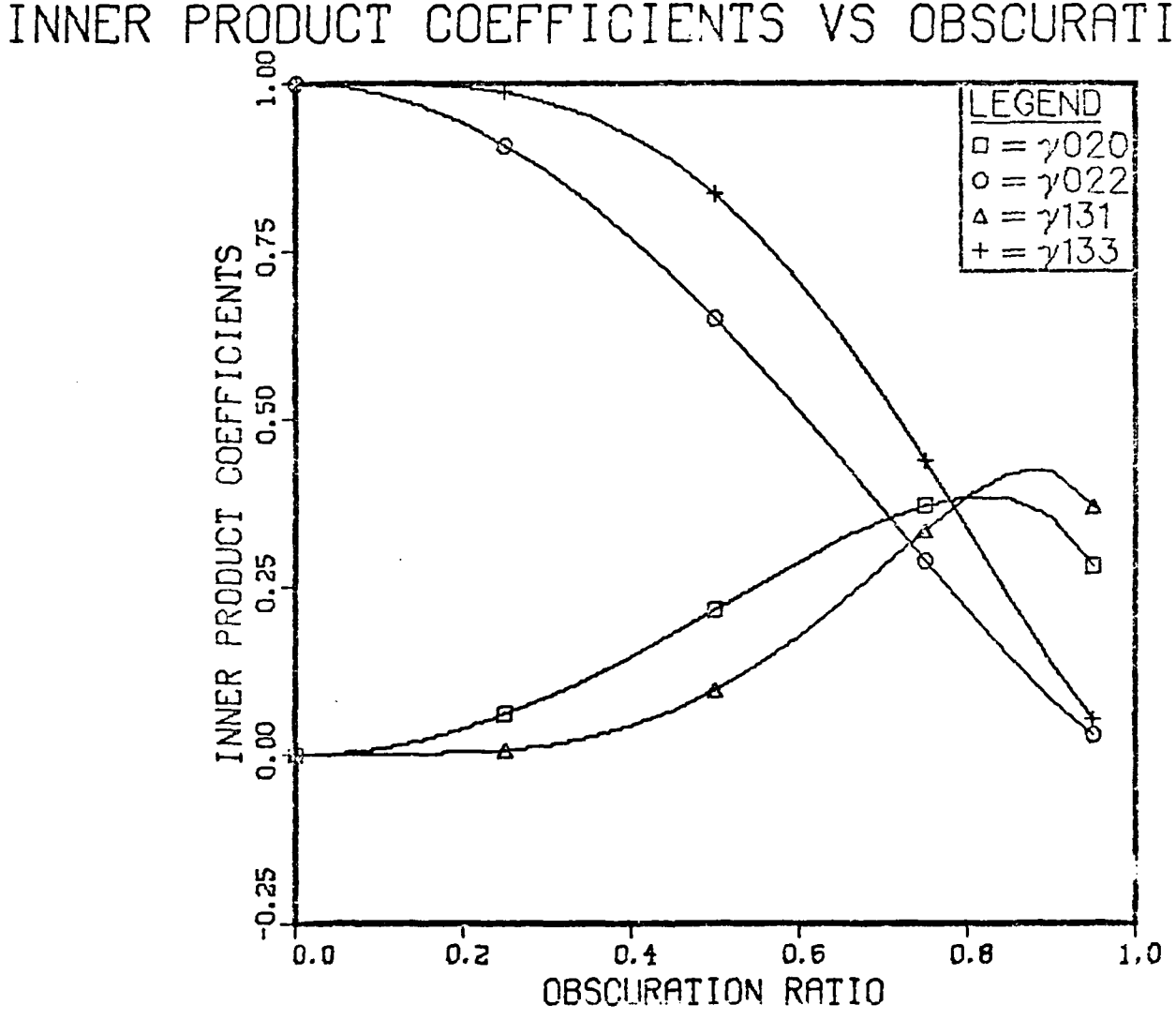


Figure 8. Inner Product Coefficients

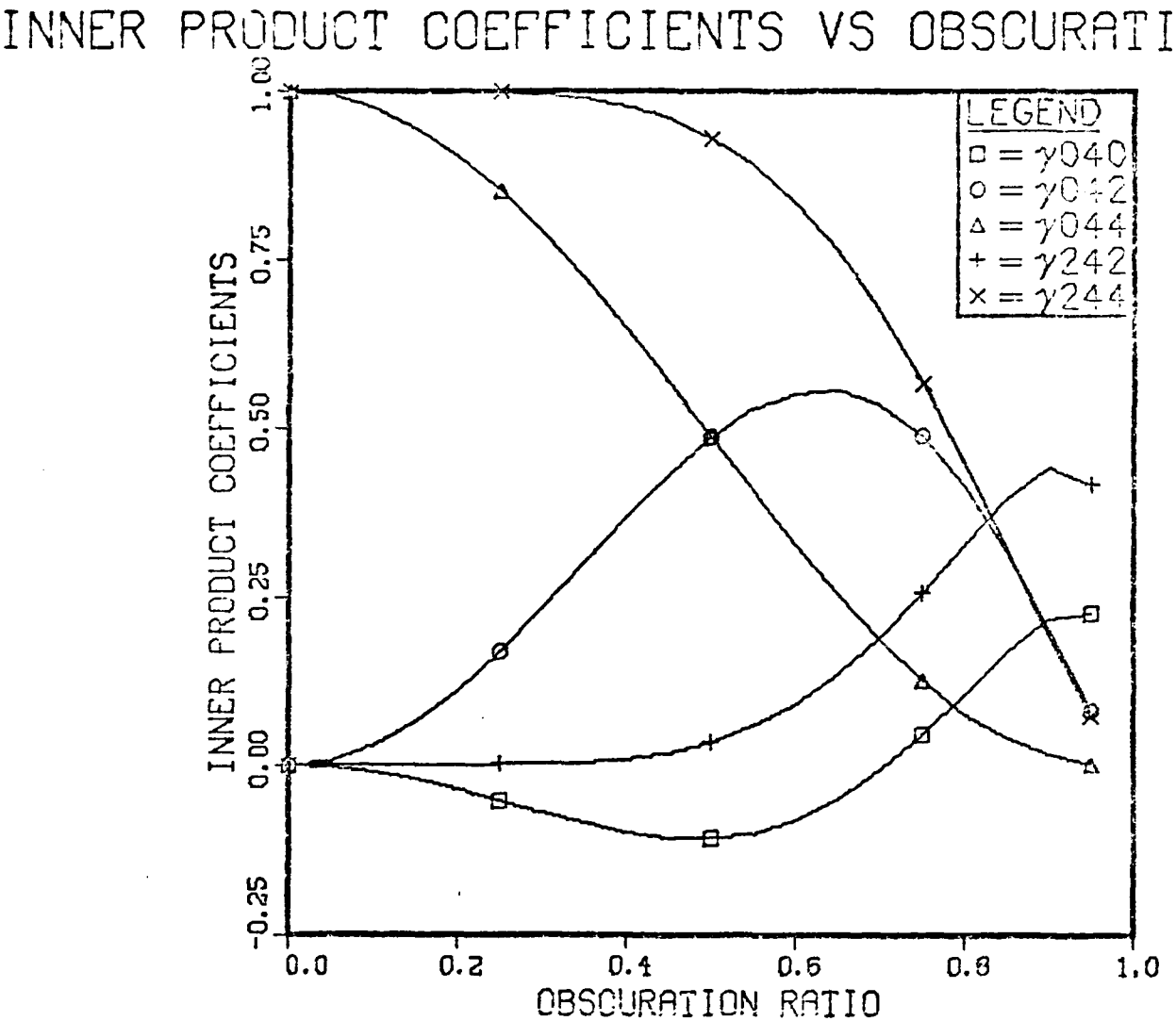


Figure 9. Inner Product Coefficients

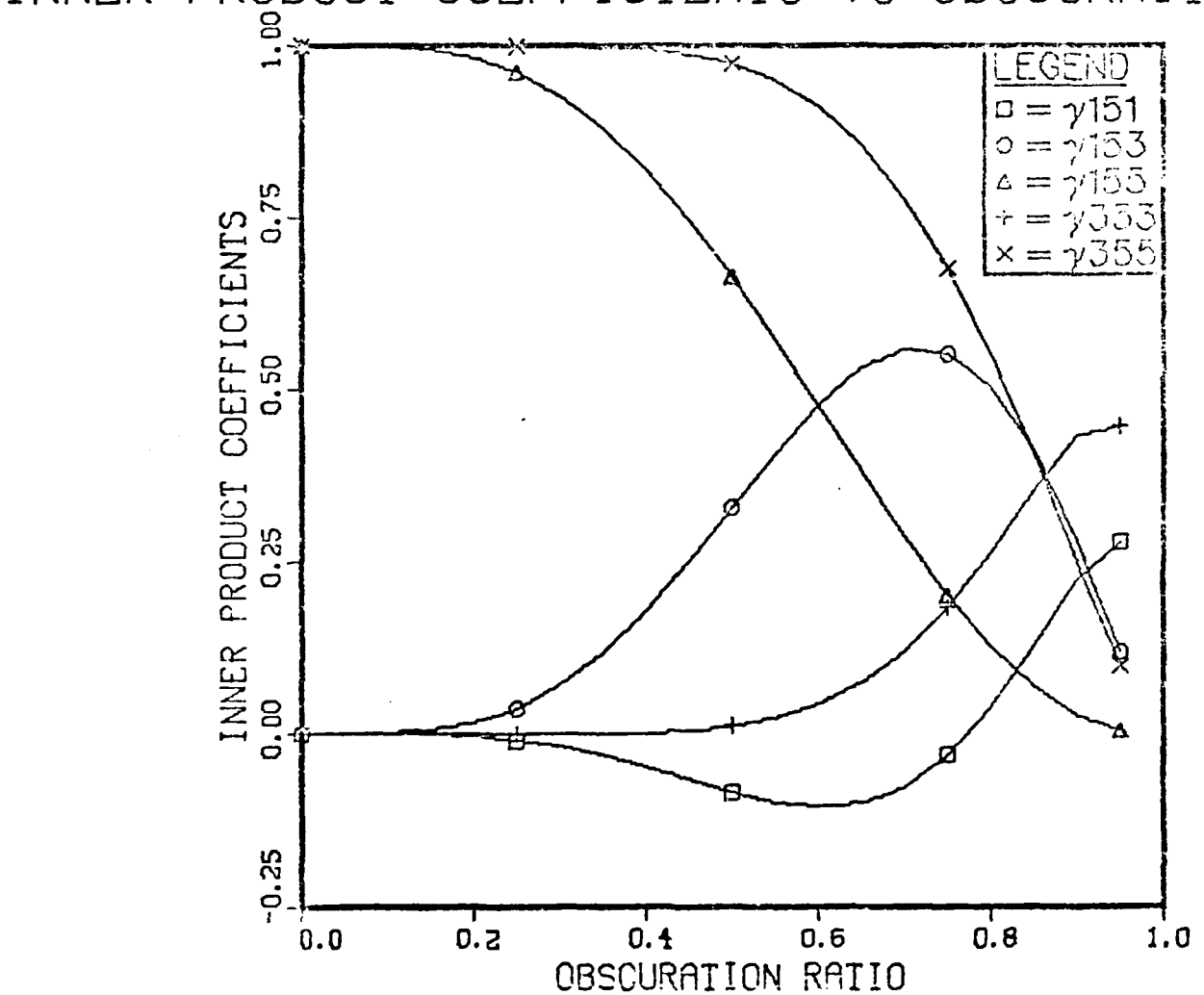


Figure 10. Inner Product Coefficients

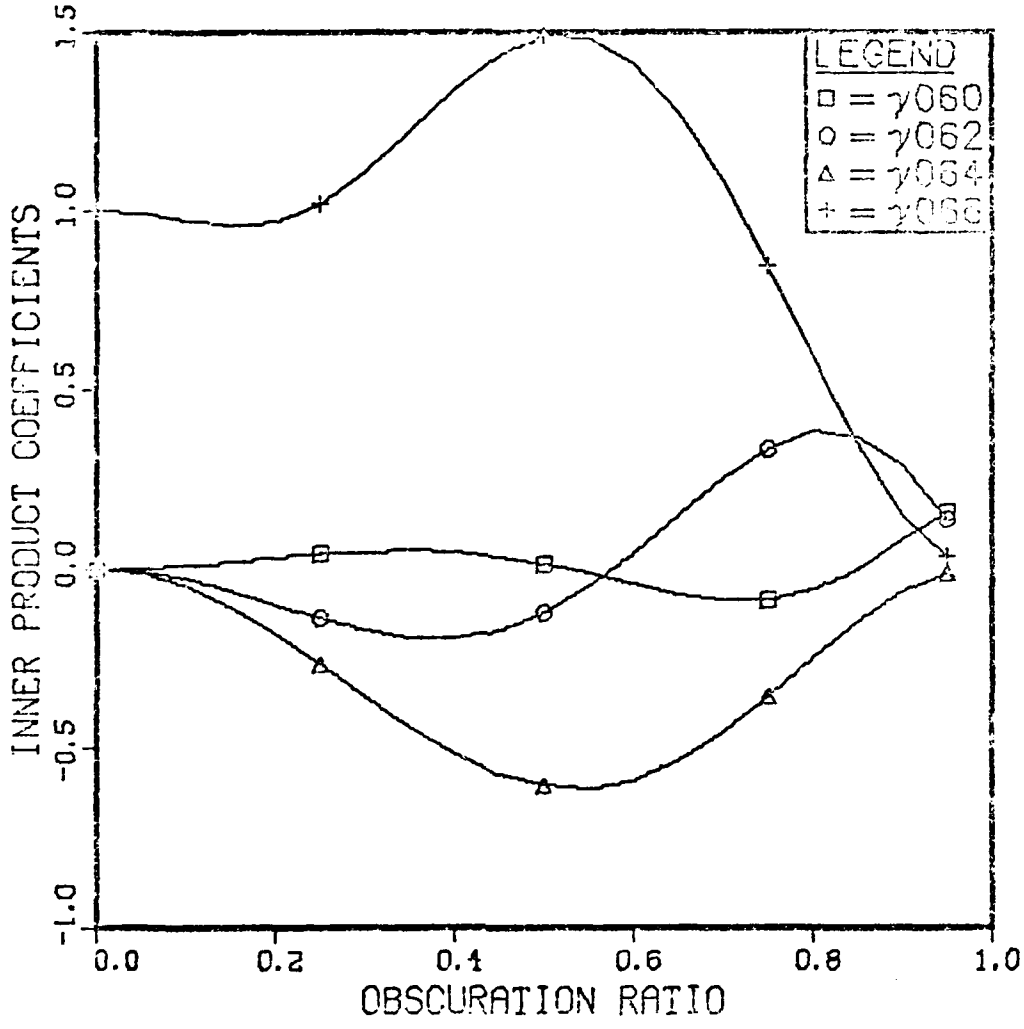


Figure 11. Inner Product Coefficients

$$\sum_{j=1}^{N} a_j Z_j(\vec{r}) \approx \phi \qquad (3.6.2)$$

As pointed out in the last chapter, ϕ is not a continuous function, but is sampled at discrete points. Thus Equation (3.6.2) can be further modified to

$$\sum_{j=1}^{N} a_j Z_j(\vec{r}_i) \approx \phi(\vec{r}_i) \qquad (3.6.3)$$

where $i=1,2,\ldots,M$, and M is the number of sampled points. This equation can be written in the matrix form

$$\underset{\sim}{Z}(\vec{r}_i)\underline{a} \approx \phi(\vec{r}_i) \qquad (3.6.4)$$

where as before $\underline{a}$ is a N by 1 column matrix, $\underset{\sim}{Z}(\vec{r}_i)$ is a 1 by N row matrix, and $\phi(\vec{r}_i)$ is a scalar.

The most common method of solving for the coefficients is to use a least-squares approach. This involves minimizing the sum of the squares of the difference between the estimated and the measured wave-fronts, or

$$\Delta = \sum_{i=1}^{M}\left[\sum_{j=1}^{N} Z_j(\vec{r}_i)a_j - \phi(r_i)\right]^2 \qquad (3.6.5)$$

To find each coefficient, the derivative of Δ is taken with respect to each a_j and set to zero. Thus

$$\frac{\partial \Lambda}{\partial a_k} = \sum_{i=1}^{M}\left[2Z_k(\vec{r}_i)\left[\sum_{j=1}^{N} Z_j(\vec{r}_i)a_j - \phi(\vec{r}_i)\right]\right] = 0 \qquad (3.6.6)$$

or

$$\sum_{i=1}^{M} Z_k(\vec{r}_i) \sum_{j=1}^{N} Z_j(\vec{r}_i)a_j = \sum_{i=1}^{M} Z_k(\vec{r}_i)\phi(\vec{r}_i) \qquad (3.6.7)$$

This equation can be written in matrix form as

$$\underset{\sim}{Z}^T\underset{\sim}{Z}\,\underline{a} = \underset{\sim}{Z}^T\phi \qquad (3.6.8)$$

and following matrix inversion and multiplication one gets

$$\underline{a} = (\underset{\sim}{Z}^T\underset{\sim}{Z})^{-1}\underset{\sim}{Z}^T\phi \qquad (3.6.9)$$

The problem with this method for finding the coefficients is the matrix $\underset{\sim}{Z}^T\underset{\sim}{Z}$ can be numerically unstable; therefore, the correct solution of the equation is difficult, but not impossible to find (Ref.8:1514).

The method used in this thesis to find the coefficients of the polynomials uses the orthogonality of the polynomials. As before, the least-squares method is used giving

$$\Lambda = \sum_{i=1}^{M}\left[\sum_{j=1}^{N} b_j N Z_j(\vec{r}_i) - \phi(\vec{r}_i)\right]^2 \qquad (3.6.10)$$

where b_j is the new polynomial's coefficient. Equation

(3.4.23) can be approximated in discrete form as

$$\sum_{i=1}^{M} NZ_u(\vec{r}_i)NZ_v(\vec{r}_i) = \delta_{uv} \qquad (3.6.11)$$

Taking the derivative of Equation (3.6.10) and setting it equal to zero gives

$$\frac{\partial \Lambda}{\partial b_k} = \sum_{i=1}^{M}\left[NZ_k(\vec{r}_i)\phi(\vec{r}_i) - NZ_k(\vec{r}_i)\sum_{j=1}^{N} b_j NZ_j(\vec{r}_i)\right] = 0 \qquad (3.6.12)$$

Since $NZ_k(\vec{r}_i)$ is a constant with respect to the summation over j, it can be put inside the second summation. Using Equation (3.6.11), the sum from $j=1$ to N is non-zero only when j equals k; therefore, Equation (3.6.12) becomes

$$b_k = \frac{1}{M}\sum_{i=1}^{M} NZ_k(\vec{r}_i)\phi(\vec{r}_i) \qquad (3.6.13)$$

With this equation, the coefficients can be found directly with no matrix inversion. Notice the same is true for the Zernike polynomials when the region is circular.

From this section and the previous one it has been shown that it is possible to construct a set of new polynomials which are orthogonal over an annular region. These new polynomials can be used to find the coefficients to

describe an annular wave-front without employing the standard matrix inversion method. This makes the programing much more reliable and faster. Since the new polynomials are derived from the Zernike polynomials, the coefficients can be transposed into the corresponding Zernike coefficients. This process will be explained in the next chapter which deals with the process of developing software to utilize the results of this chapter.

IV. Program Development

This chapter explains the development of software which performs the analysis of annular cave-front using the theory presented in the last chapter. The software consists of two major sections, the definition of the region of analysis and the actual analysis of the wave-front. This chapter presents the main program and each of the major subroutines, with the subroutines being presented in the order in which they are called. Each section will present the basic algorithm, a basic flowchart, and a description of the algorithm as necessary.

Main Program

The main program is the controlling program of most of the subroutines. It does the actual analysis of the wave-front and prints the results for the user. The basic algorithm for this program is:

1. Read in first frame of data and call FAPER. FAPER returns the computed radius, center, and obscuration ratio. The operator is asked if he wants to change any parameter and the number of frames to be analyzed.

2. If not the first time through the routine, then read in a new frame of data.

3. Define region of analysis as given from parameters computed or entered in step 1 by calling CONTUR.

4. Do steps 5 to 11 three times: once to compute 6 coefficients, then to compute 11 coefficients, then to compute 22 coefficients.

5. Zero out all data arrays used in computation.

6. Using the obscuration ratio, compute the inner product coefficients by calling GAMSUB.

7. Compute X and Y starting positions and increments.

8. Using the region defined in step 7 and the parameters from step 4, compute the new polynomial coefficients.

9. Convert coefficients from step 8 to Zernike coefficients.

10. Generate two estimated wave-fronts from the Zernike and new polynomial coefficients.

11. Compute RMS error and print results.

12. If not done with all frames do to step 2.

13. Stop.

This algorithm is presented in flowchart form in Figure 12. Since each subroutine is presented later, only those steps which do not rely on major subroutines are presented here.

The first step in the algorithm uses FAPER to define the wave-front. After this is done, the user is asked if he wants to see the defined region and the computed parameter. These parameters include the center of the annular region, the outside radius, and the obscuration ratio. The user is then asked if he wants to change any parameters if he knows the actual values. This request is made to correct any possible errors that FAPER may have made. The reasons for error in FAPER will be presented later. The program will also ask the user how many frames he wants to have

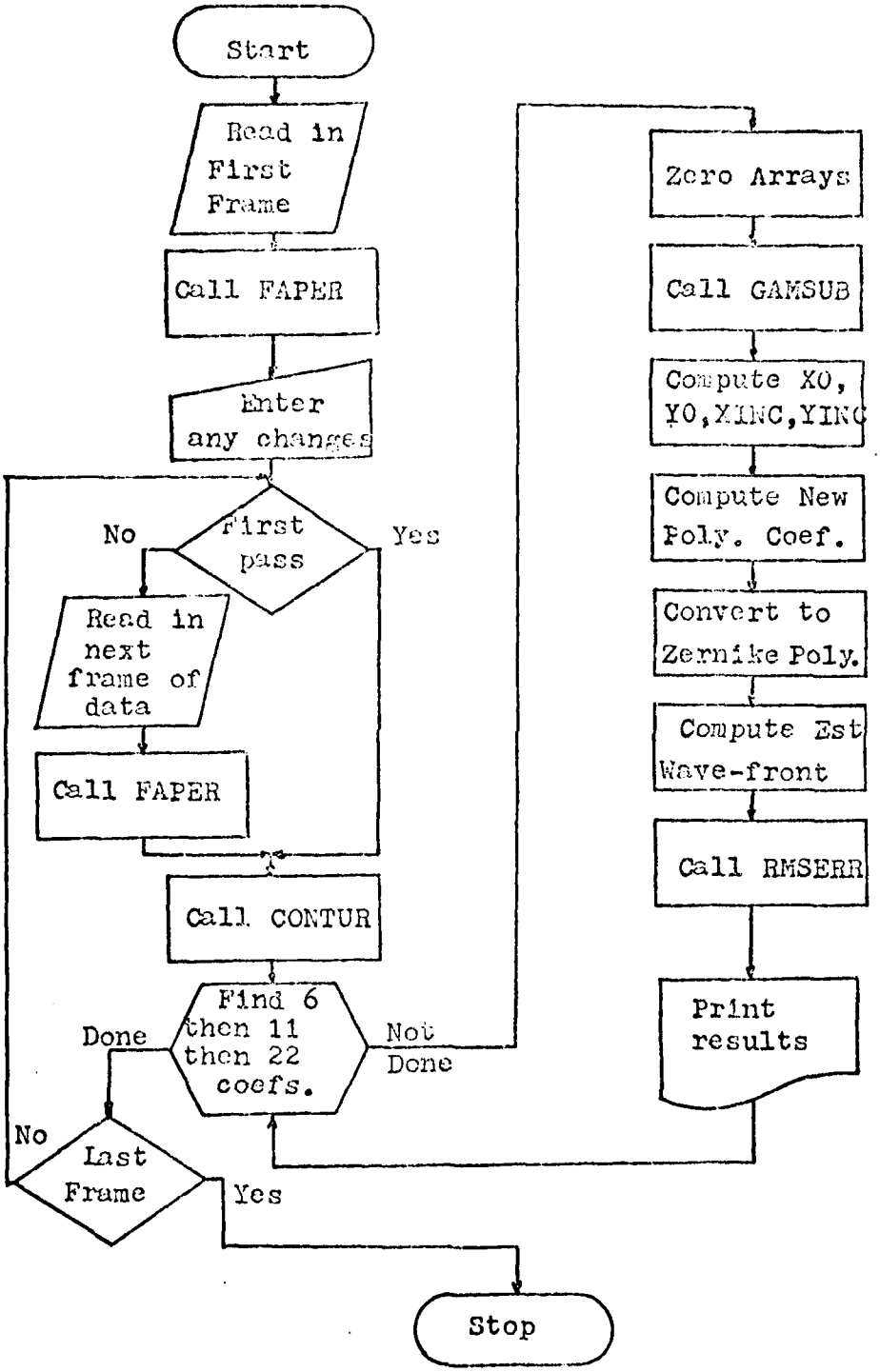


Figure 12. Basic Flowchart of Main Program

analyzed. With this data, the program is ready to start the actual analysis.

After the first frame has been analyzed, a new frame is read. This frame of data is defined into regions of no data, valid data, and the center of the annulus by the parameters determined from the first frame. This is done by the subroutine CONTUR. Once the regions of the wave-front are defined, the program starts to analyze the wave-front. Each wave-front has 6, then 11, then 22, coefficients computed to verify that the number of coefficients does not affect the values of the coefficients. Before each new set of coefficients is computed, the data arrays are zeroed.

Using the obscuration ratio parameter, the subroutine GAMSUB is called. This subroutine computes the inner product coefficients needed to convert Zernike radial polynomials to the new set of radial polynomials. The resulting lower triangular matrix GAMMA is inverted and placed into INGAMMA for future use. Each array is used in the conversion process depending on which radial polynomial is being converted to the other. With these arrays, the program can begin to analyze the phase data.

Since the phase array is not always filled, the range of X and Y depend on the size of the radius. In other words, if the radius is only ten units in a 32 by 32 unit

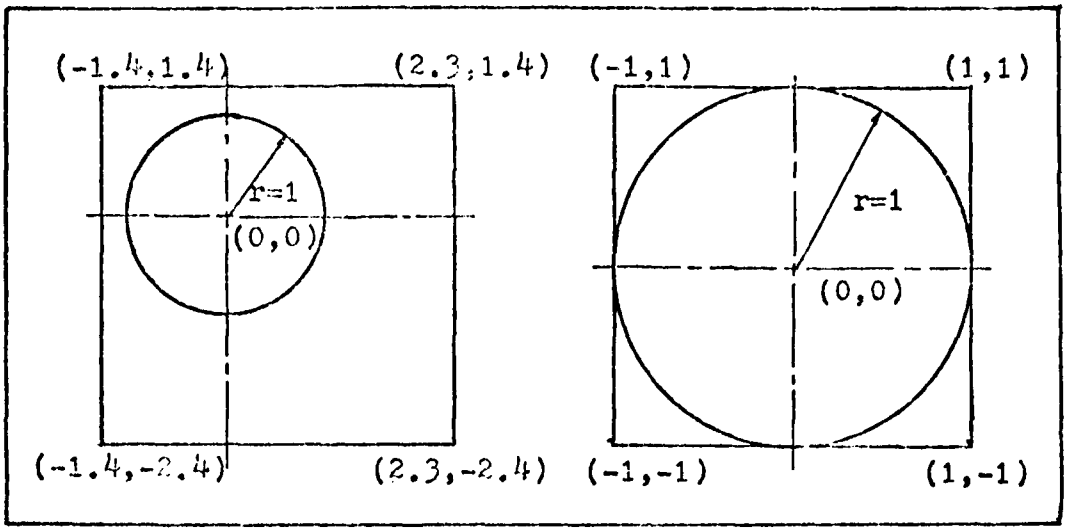


Figure 13. Possible Cartesian Coordinates of Phase Array

array, the position of the wave-front will change the X and Y values of the four corners. This can be seen in Figure 13.

Using the radius and center, the program must determine the X and Y increment and the initial values XO and YO . The analysis starts in quadrant II of the coordinate system, and scans from -X to X , Y to -Y ending in quadrant IV . XO and YO are found, such that

$$XO = \frac{1-X}{\text{Radius}} \qquad (4.1.1)$$

and

$$YO = \frac{Y-1}{\text{Radius}} \qquad (4.1.2)$$

where X and Y are the center coordinates of the wave-front given in terms of increments in the 32 by 32 array.

Thus the point (1,1) in the array corresponds to the point (-XO,YO) in Cartesian coordinates. The increment values for X and Y are simply found by taking the reciprocal of the radius of the wave-front.

The first step in the analysis is to compute the coefficients to the new set of polynomials. As pointed out in the last chapter, each new radial polynomial is made up of a linear combination of Zernike radial polynomials. This can be expressed in matrix form as

$$\underline{\gamma}\underline{NZ} = \underline{R} \qquad (4.1.3)$$

where $\underline{\gamma}$ is a square, lower triangular matrix, $\underline{NZ}$ is the new set of radial polynomials, and $\underline{R}$ is the Zernike radial polynomials. It was pointed out that either of the radial polynomials are used with the angular dependence to construct the total polynomial; therefore, the program computes the angular data and multiplies it times either radial polynomial to generate the total polynomial. Thus to compute the new polynomial, the following matrix equation is used,

$$\underline{NZ} = \underline{C}\gamma^{-1}\underline{R} \qquad (4.1.4)$$

where $\underline{C}$ is a diagonal array of angular constants. As an example the equation can be expressed as

$$\begin{bmatrix} NZ_1(r) \\ NZ_2(r) \\ NZ_3(r) \\ NZ_4(r) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2\cos\theta & 0 & 0 \\ 0 & 0 & 2\sin\theta & 0 \\ 0 & 0 & 0 & \sqrt{3} \end{bmatrix} \begin{bmatrix} 1/\gamma_{00}^{0} & 0 & 0 & 0 \\ 0 & 1/\gamma_{11}^{1} & 0 & 0 \\ 0 & 0 & 1/\gamma_{11}^{1} & 0 \\ -\gamma_{20}^{0}/\gamma_{22}^{0} & 0 & 0 & 1/\gamma_{22}^{0} \end{bmatrix} \begin{bmatrix} R_1(r) \\ R_2(r) \\ R_3(r) \\ R_4(r) \end{bmatrix}$$

where the first four new polynomials are desired.

To actually compute the coefficients, the program will scan the phase array. As soon as the radial value on the point in the scan is less than one, $\sqrt{x^2+y^2} \leq 1$, the Zernike radial polynomials $\underline{R}$ and the angular constants $\underline{\underline{C}}$ are computed and the matrix multiplication indicated in Equation (4.1.4) is performed. Using Equation (3.6.13), the coefficient for each polynomial is computed. Once the entire phase array is scanned, the coefficients are found by dividing the summation by the total number of points used in determing the coefficient. This process can be expressed as

$$b_k = \frac{1}{M} \sum_{i=1}^{M} NZ_k(\vec{r}_i)\phi(\vec{r}_i) \qquad (4.1.5)$$

where, as before, b_k is the coefficient to the new polynomial $NZ_k(\vec{r}_i)$, and M is the number of valid data points. A valid data point is one which is inside the annulus or unit circle, and is a phase measurement. Thus, using Equation (4.1.4) and (4.1.5), the new polynomial coefficients are determined. The next step is to convert these coefficients to

Zernike polynomial coefficients.

The Zernike coefficients can be easily found by first converting the new polynomial coefficients to coefficients of the new radial polynomials. Next the radial coefficients are converted to Zernike radial coefficients and finally to Zernike polynomial coefficients. This process can be expressed as

$$\underline{a} = \underline{C\gamma C^{-1}}\,\underline{b} \tag{4.1.6}$$

where $\underline{b}$ is the new polynomial coefficients, and $\underline{a}$ is the Zernike polynomial coefficients. As before, the process is done be scanning through the array and calculating the values of $\underline{C}$ and $\underline{C}^{-1}$ at each valid point. Each coefficient is kept as a running number of valid points at the end of the scan. Equation (4.1.6) can be expressed as

$$a_k = \frac{1}{M}\sum_{i=1}^{M} C_k(\vec{r}_i)\gamma_k C_k^{-1}(\vec{r}_i)b_k \tag{4.1.7}$$

With the two sets of coefficients, the next step is to compute the estimated wave-fronts.

Reconstructing the estimated wave-front is basically the opposite of finding the coefficients. The only difference is that the two wave-fronts can be found at the same time. The matrix equations are

$$\phi_z = \underline{a}\ \underline{C}\ \underline{R} \tag{4.1.8}$$

and

$$\phi_{nZ} = \underline{b}C_Y^{-1}\underline{R} \qquad (4.1.9)$$

where ϕ_Z and ϕ_{nZ} are the wave-fronts found with Zernike and new polynomial coefficients respectively. The wave-fronts are formed by scanning through the 32 by 32 array and at each valid point, the matrix Equations (4.1.8,9) are evaluated. Thus after the scan, there are two estimated wave-fronts.

Once the estimated wave-fronts are constructed, the Root Mean Square (RMS) error is calculated. This error is found by calling the subroutine RMSERR. Its operation is explained later. Once the RMS error is found, the main program will print out the coefficients of both polynomials and the RMS error from their respective estimated wave-fronts. The program will then either return to compute the next set of coefficients, or start a new frame of data.

In summary, the main program does the actual analysis of the wave-front. It also asks for any input parameters and calls various subroutines to prepare the frame for analysis. Appendix D contains detailed flowcharts of this routine and all the subroutines. The next section deals with the subroutine which computes the inner product coefficients.

GAMSUB

The subroutine GAMSUB is a routine which computes the inner product coefficients, γ_{nj}^{m} . This routine has three parameters in its calling statement: the array to hold the inner product coefficients, the obscuration ratio, and the number of coefficients.

As pointed out in the last chapter, the actual inner product terms are dependent on each other. To make the program run faster and more accurately, each of the integrals in Equations (3.5.7,17) were done by hand. This results in a series of dependent equations with a dependency on lower order terms and the obscuration ratio, β . Since the integrations where done by hand, each integral does not have to be done by some numerical method, i.e. Simpson's rule or the trapezoidal method. This greatly reduces the time since only one pass through the routine is required instead of integrating each term. The error is reduced by not approximating a term which is used in the approximation of another term.

Since this subroutine is just a series of equations, the algorithm will not be presented. GAMSUB's flowchart is presented in appendix D, and the entire matrix it generates is in appendix E. Before GAMSUB can run, the obscuration ratio must be found. It is found in the subroutine FAPER which is presented in the next section.

FAPER

The subroutine FAPER returns the center, outside radius, and the obscuration ratio. It also defines the value for no data present. This subroutine is the most important one since any errors can affect where the main program "thinks" the wave-front is positioned. The main purpose of this subroutine is to define the type of wave-front, i.e. annular or circular, and then find the parameters.

Since most of the analysis will be with annular wave-fronts, FAPER will assume an annular wave-front is present. The algorithm for this subroutine is:

1. Select the value to be used as no data (BAD) as the value which is present at any three corners of the collection array.

2. If the point (16,16) in the array contains a value of BAD, go to step 5.

3. If not, starting at (16,16), spiral outward until a point with a value of BAD is found or the radial distance from (16,16) is greater than 5.75.

4. When a BAD point is found, set new center to coordinates of this point. If the distance was greater than 5.75, then keep the center at (16,16).

5. By calling EDGE, find the vertical and horizontal radii of the inside radius of the annulus. If EDGE returns an error condition, the wave-front is non-annular; therefore go to step 6. Repeat until the radii stabilize or ten iterations have been done.

6. From the center found in step 4, find the smallest outside radius.

7. If the region was defined as non-annular, define the wave-front by calling CONTUR, and find the largest outside radius which will fit inside the wave-front. Pass this value and the center coordinates to VALID to

find the floating-point values of the radius and center.

8. If the region is annular, use CONTUR to define the region. VALID is called to find the smallest inside radius for the annulus which will circumscribe the BAD data. The inside radius and the smallest outside radius is used to define the largest possible outside radius.

9. Fill the character array with final symbols.

10. Return to calling routine.

The basic flowchart for this subroutine is shown in Figure 14.

The first two steps of the algorithm are very straight forward. The determining of the value of no data is crucial to the entire program's operation. This and other subroutines use this value as a test to determine whether to use a point in the analysis of the wave-front. It is also used to define the central region of the annulus. The next step is to find, and if possible, define the inner radius of the annulus.

When the LWA is run, the input wave-front is usually centered on the collection array; therefore, this subroutine will start looking at the center for the annulus. If a valid data point is present at the center (16,16), then the subroutine will spiral outward to try to find the central region. Figure 15 shows the shape and direction of the spiral at the largest extent of the search.

If the data at any point along the search is invalid, the program will assume it has found the edge of the inside

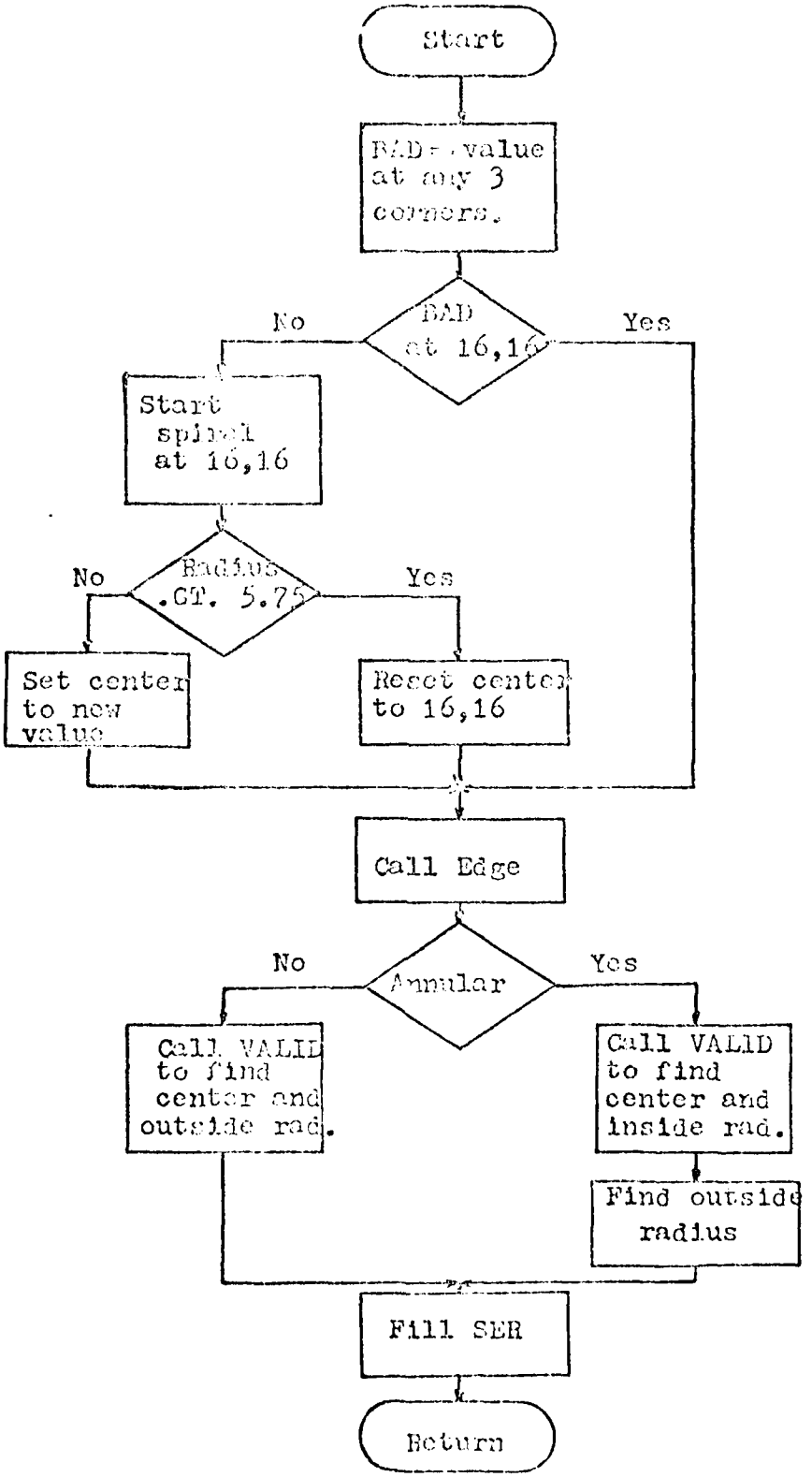


Figure 14. Basic Flowchart of FAFER

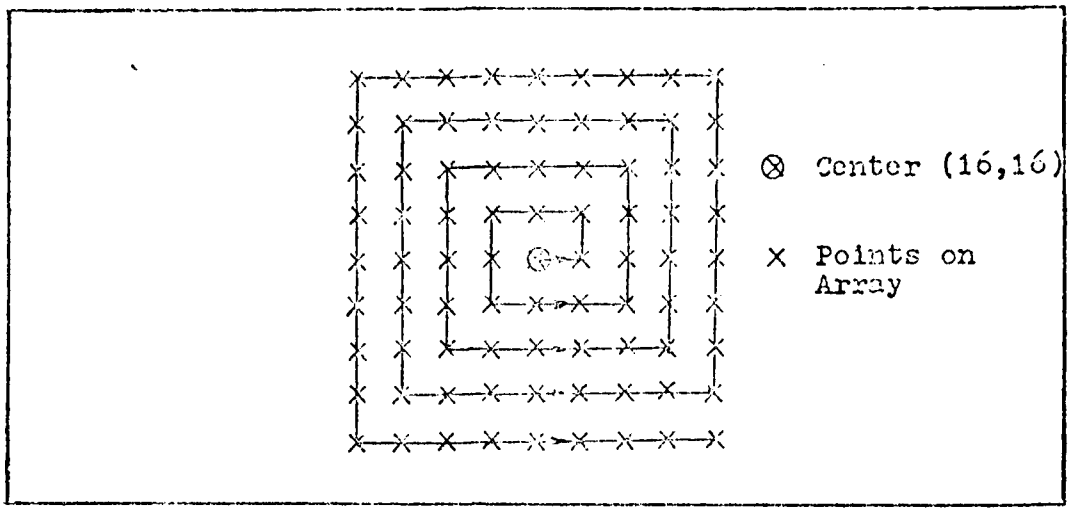


Figure 15. Spiral Search Pattern of FAPER

region of the annulus. Invalid means that the point has a value of BAD. The maximum radius of 5.75 for the search was chosen as an optimum value from several test cases. The entire search process is based on the assumption that the operator of the LWA has aligned the system to fill the array. If an invalid point is found in the search, this point is tentatively set as the new center of the annulus.

The next step is to find the approximate center of the annular region. Assuming the wave-front is annular, the subroutine EDGE is called to find the horizontal and vertical radii from the center in all four directions. Using the averages of the two horizontal and two vertical radii, a new center is computed and EDGE is called again. This process is repeated until either the radii do not

change or ten iterations are performed. The limit of ten iterations is to prevent the possibility of the integer center oscillating around the actual center. If EDGE runs into the limits of the array, then FAPER will assume the spiral search found the outside edge of a circular wave-front. If this happened, the center is reset to (16,16). At this point the program "knows" either the approximate location of the center of the annulus or that the wave-front is non-annular.

In either type of wave-front, the next step is to find the smallest horizontal and vertical radii from the center. With this radius, in the case of an annulus, VALID is called to find the center and the smallest radius which covers the inside region of the annulus. When the wave-front is non-annular, the radius is used to find the largest radius. VALID is again called to look for the center and the largest circle which will fit in the region.

If the wave-front is non-annular, then FAPER returns the results of VALID to the calling routine. Otherwise FAPER will search for the outside radius of the annulus. This is done by using the floating-point values of the center found by VALID. This radius and the inside radius are used to compute the obscuration ratio, BETA. BETA, the center, and the outside radius are then returned to the calling routine.

In summary, PAPER is the controlling routine to define the type and parameters of the wave-front. This routine is called to show the operator what it "thinks" the wave-front looks like. With this information, the operator can accept or reject its results; thereby reducing the likelihood of error. The next section deals with the subroutine VALID.

VALID

The subroutine VALID's main purpose is to compute the floating-point value of the center and the radius of the wave-front. As pointed out in the previous section, VALID has two possible modes of operation: find the center and outside radius of a non-annular wave-front, or find the center and inside radius of an annular wave-front. The basic algorithm is:

1. Using the flag passed to it, go to step 3 if a non-annular wave-front.

2. Define the region of search to the center ± the estimated radius plus one. Go to step 4.

3. Define the search area as the entire array.

4. Search the defined region for either valid or invalid points depending on whether the wave-front is non-annular or annular.

5. Keep a running total of the number of points found, and the values of the vertical and horizontal indices if looking for the center.

6. Divide the sums of the vertical and horizontal indices by the total number of points, to get the true center of the region.

7. Using the new center and the estimated radius,

repeat the search. If annular region, count the number of invalid points, decreasing the radius of search by 0.25. As soon as the new count changes from the previous count, use the last value of the radius as the smallest radius. If non-annular, increase the radius until an invalid point is found and use previous radius as the largest radius.

8. In non-annular, return the new center and the radius to the calling routine.

9. If annular, fill character array SER with the symbol for the central region of the annulus and return the calling routine.

A flowchart of this routine is shown in Figure 16.

As indicated in the algorithm, and the previous section, VALID is passed the integer estimate of the center and radius from the calling routine. It uses these values to try to find the floating-point values of the same. To avoid confusion, the algorithm will be split into the two cases of annular and non-annular analysis. In the case of an annular wave-front, VALID will use the integer radius to circumscribe the inside region of the annulus with the search region. On the first pass through, VALID will compute the first moments of the X and Y values of the coordinates of the invalid data points. The moments are found by using the following equations

$$XC = \frac{1}{NT} \sum_{i=1}^{N} \nu_i X \qquad (4.4.1)$$

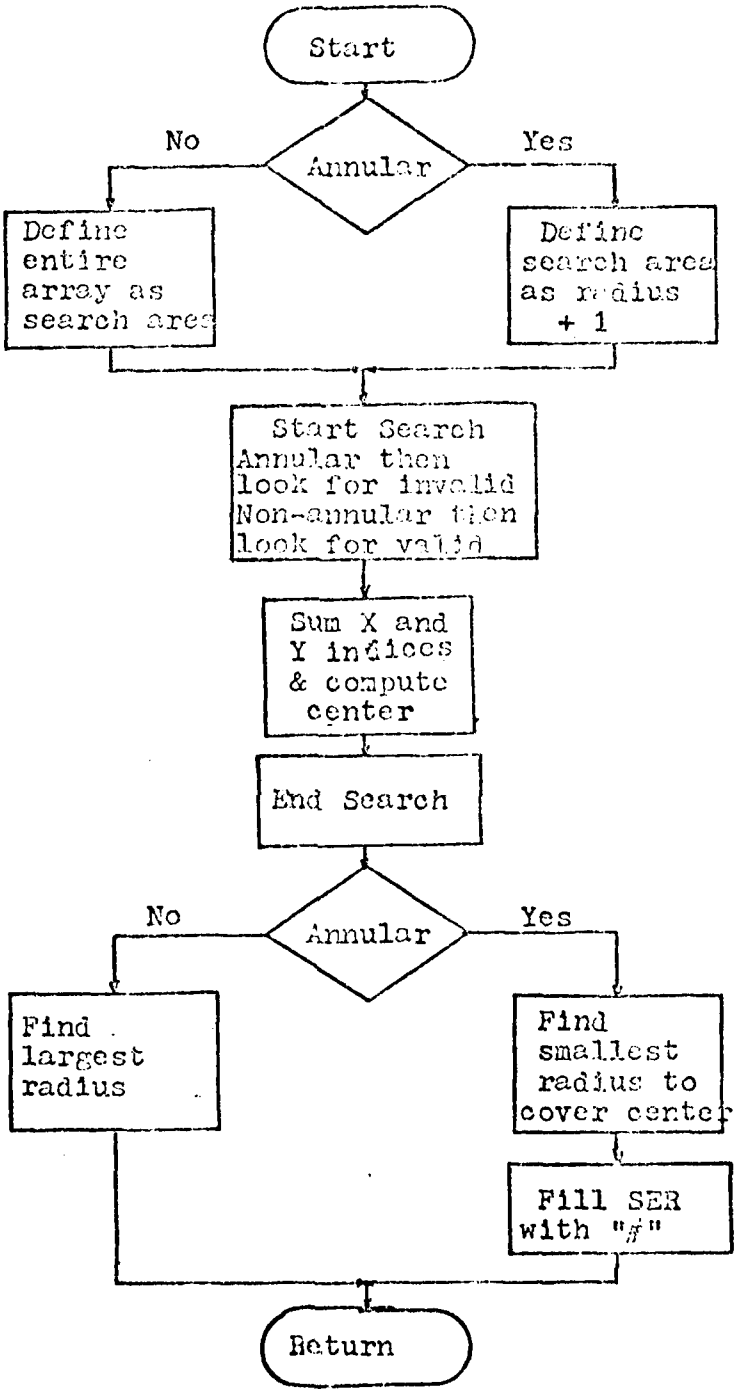


Figure 16. Basic Flowchart for VALID

and

$$YC = \frac{1}{NT} \sum_{i=1}^{N} \nu_i Y \qquad (4.4.2)$$

where

$$\nu_i = \begin{cases} 1 \text{ when the data is invalid} \\ 0 \text{ otherwise} \end{cases}$$

NT is the number of invalid points in the search area, and N is the number of points in the entire search area. XC and YC in these equations are the floating-point values of the center. Once the center is found, the search is repeated and the smallest radius is found which will include all of the invalid data points (NT). The radius is changed in steps of 0.25. This decrement was selected because the use of a floating-point radius and center in a discrete array gives a reasonable decrement without sacrificing accuracy. Once the radius is found, VALID fills the character array SER with the symbol "#" to represent the center of the annulus. VALID then returns its results to the calling routine.

In the case of a non-annular r gion, VALID defines the search area as the entire data array. It then follows the same procedure as before. VALID first finds the center by using the first moments as in Equations (4.4.1,2) but

$$\nu_i = \begin{cases} 1 \text{ when the data is valid} \\ 0 \text{ otherwise} \end{cases}$$

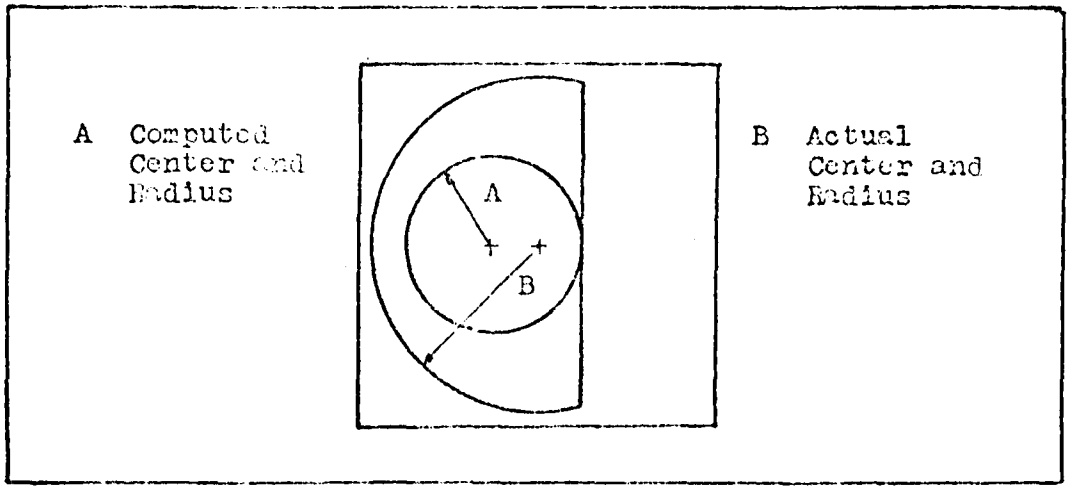


Figure 17. Error Caused by VALID's Means of Finding Center

Using the computed center and the estimated outside radius, VALID will search the array. If an invalid point is found in the radius of search, the radius is decreased by 0.25. This repeated until no invalid points are found; thus, the smallest radius is found which circumscribes the valid data. The radius and center are then returned to the calling routine.

Since VALID uses the first moment to find the center, errors can be introduced. Figure 17 shows one such example where the input beam is a semicircle. This is a case where user intervention is important. The user can reset the center and radius parameters to best fit the wave-front. VALID works best with wave-fronts which are circularly symmetric. The further the wave-front is from being symmetric, the greater the possibility of error.

The subroutine VALID is used to find the floating-point values of the center and radius, from integer guesses. It works best with wave-fronts which are symmetric. VALID can give erroneous data; therefore, the main program will check with the user before continuing.

EDGE

The subroutine EDGE is called by FAPER to estimate the center of the annulus. EDGE has two modes of operation: vertical or horizontal scan. The mode is selected by a flag which is passed to EDGE. The algorithm for EDGE is:

1. Starting at coordinates given by calling routine, search left or down until a valid point is found. Save the distance.

2. Starting at the same coordinates, search right or up until a valid point is found. Save the distance.

3. If the search in any direction hits the outside boundary of the data array, then set error flag and return. Otherwise return left/right or up/down distances to calling routine.

The basic flowchart for this algorithm appears in Figure 18.

As pointed out in the section on FAPER, EDGE is called at the most ten times to try to find the best guess for the center of the annulus. The distance returned is also used in the determination of the inside radius. The next sections deal with subroutines which are used in the analysis of the wave-front.

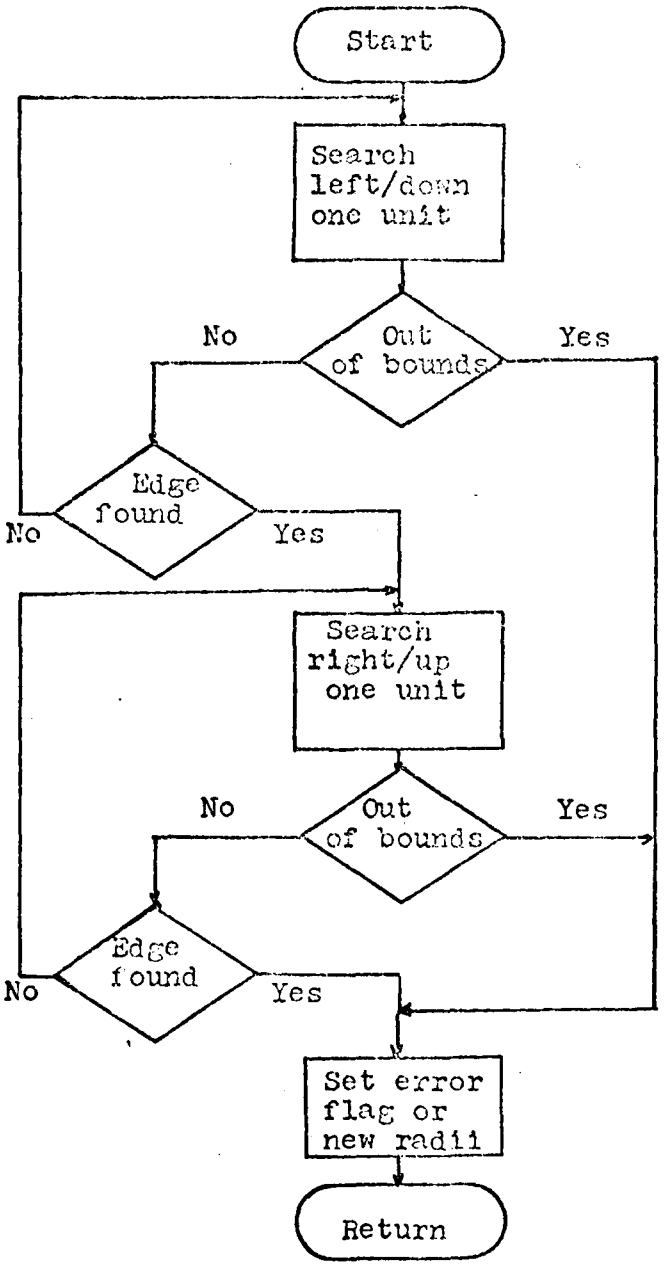


Figure 18. Basic Flowchart for EDGE

ZRAD

The subroutine ZRAD is used to generate the values of the Zernike radial polynomials at a specific X , Y , coordinate. Equations (4.1.4,5) show that to find the coefficients to new polynomials, the new polynomials must be computed at each X , Y point. ZRAD is a very simple subroutine, as shown in the algorithm. The algorithm is:

1. Using the X,Y values and the number of terms needed, compute the Zernike radial polynomial terms $R_n^m(X,Y)$.

2. Fill a column matrix with the desired number of terms.

ZRAD uses the radial polynomials presented in Table II. As an example when six coefficients are needed, ZRAD will produce a column matrix consisting of $\{R_0^0(X,Y), R_1^1(X,Y), R_1^1(X,Y), R_2^0(X,Y), R_2^2(X,Y), R_2^2(X,Y)\}$. The repetition of some of the terms comes from Equation (3.2.2). The index of the matrix $\underline{R}$ corresponds to j in Equation (3.2.2). The entire matrix generated by ZRAD is presented in appendix E. Thus ZRAD is a simple but necessary routine for the generation of either Zernike or new polynomials.

ZANG

The subroutine ZANG is similar to ZRAD. ZANG computes the angular dependence of the Zernike polynomials for a specific X,Y coordinate. Instead of producing a column matrix, ZANG makes a square diagonal matrix. The matrix

is shown in Appendix E. The index into the array, (j,j) where j=1 to the number of coefficients, is the mode-order j in Equation (3.2.2). ZANG simply computes the constant and sine/cosine portion of the Zernike polynomials. ZANG and ZRAD are separate from each other to allow easy conversions of new polynomial coefficients to Zernike coefficients as shown in Equation (4.1.6).

CONTUR

The subroutine CONTUR is used to fill a character array with symbols. CONTUR uses the center coordinates and the inside radius of the annulus to fill the array SER with: "." when the data point is valid, "*" when no data is present, and "#" for the center of the annulus.

CONTUR is a simple program which scans the phase data, putting in the proper symbol. When the scan gets inside the central region of the annulus, the "#" is used regardless of the value of the data point.

CONTUR is the last subroutine which is used in the analysis of the data. The next sections deal with either data manipulation or the output of results.

RMSERR

The subroutine RMSERR finds the normalized RMS error between two arrays. The basic equation used is

$$\text{ERROR} = \sqrt{\frac{\sum_{i=1}^{N}(A_i - B_i)^2}{N}}$$

where A_i is the input wave-front array, B_i is the estimated wave-front, and N is the total number of points used in computing the estimated wave-front. To find the number of points, RMSERR uses the "Λ" in the character array. This symbol was placed in the array by FAPER to tell the remaining routines where valid data for analysis can be found. Thus whenever a "Λ" is found, RMSERR then adds another term to the summation. The normalization allows a better comparison between obscured wave-fronts and non-obscured wave-fronts.

Minor Subroutines

The minor subroutines include SERPRNT, INVERT, MULT, ARPR1, and ARPR2. The subroucine SERPRNT is called whenever the character array SER is printed. SERPRNT also prints out the parameters used in the analysis of the wave-front. The parameters are: the center, radius, and obscuration ratio.

INVERT and MULT are two matrix routines. INVERT inverts a lower triangular, square matrix. It is INVERT which inverts the inner product coefficient array and the angular constant array. MULT multiplies a square matrix times a column matrix. MULT is used to do the matrix

multiplication in Equations (4.1.6,8,9).

ARPR1 and ARPR2 are used to debug the routines. ARPR1 prints out a column matrix and ARPR2 a square matrix. They are not called by any routine, but are left for the user to use if he decides to make any modifications.

Conclusions

This chapter has presented the main routines and all of its subroutines with each routine being explained to show how the theory of the last chapter was utilized. It was shown how the routines are able to analyze both annular and non-annular wave-fronts. Appendix D contains detailed flow-charts of all of the routines, and Appendix F contains a listing of the entire program. The software has been written such that is does not rely on any system subroutines other than a square root routine, which reduces the possibility that a special routine is not available. The next chapter deals with the validation and verification of the routines presented in this chapter.

V. Validation

The previous chapters presented the reasons for developing a new software package, the theory behind the development, and the actual software. This chapter deals with the validation of the software. The validation process is broken up into three sections. The first section tests the orthogonality of the new polynomials as the obscuration ratio changes. The next section verifies the software's ability to analyze non-annular wave-fronts. The final section deals with annular wave-fronts. Thus all of the aspects of the software will have been tested and verified.

Orthogonality Verification

The major aspect of this thesis is the generation of an orthogonal set of polynomials. The polynomials must be orthogonal over an annular wave-front. This section verifies that such a set has been generated. Since both the Zernike and new polynomials use the same angular functions, only the radial portions are tested, as shown in Equations (3.2.4) and (3.4.23).

The actual test consisted of performing the integrals in Equations (3.2.4) and (3.4.22) with the computer. The integrals were approximated by using Simpson's rule. The region of integration was divided into 3000 steps for the

initial integration. This was doubled until the difference between iterations was less than 0.01. To show a comparison between the Zernike and new radial polynomials, the first integration had limits of 0.0 to 0.99999999. The upper limit was not 1.0 because the routine GAMSUB would try to compute some indefinite terms.

The test integrated all the possible combinations of Zernike radial polynomials and all the possible combinations of the new radial polynomials. After each integration, the lower limit would increase by 0.05 until the limits of integration were 0.95 to 0.99999999. Figure 19 is a plot of the non-zero terms of the orthogonality condition. All of the terms where the Kronecker delta is supposed to be zero oscillated around zero. The value of these terms were very dependent on the number of steps in the integration approximation; therefore, only the cases where the Kronecker delta is one were plotted. Figure 19 clearly shows that the values of the new radial polynomials remained constant regardless of the integration limits.

To verify the cases where the Kronecker delta is zero, the integrations were done by hand using a variable as the lower limit. All of the integrals verified the orthogonality condition.

This test verified the inner product coefficients, the program GAMSUB, and the theory presented in Chapter III.

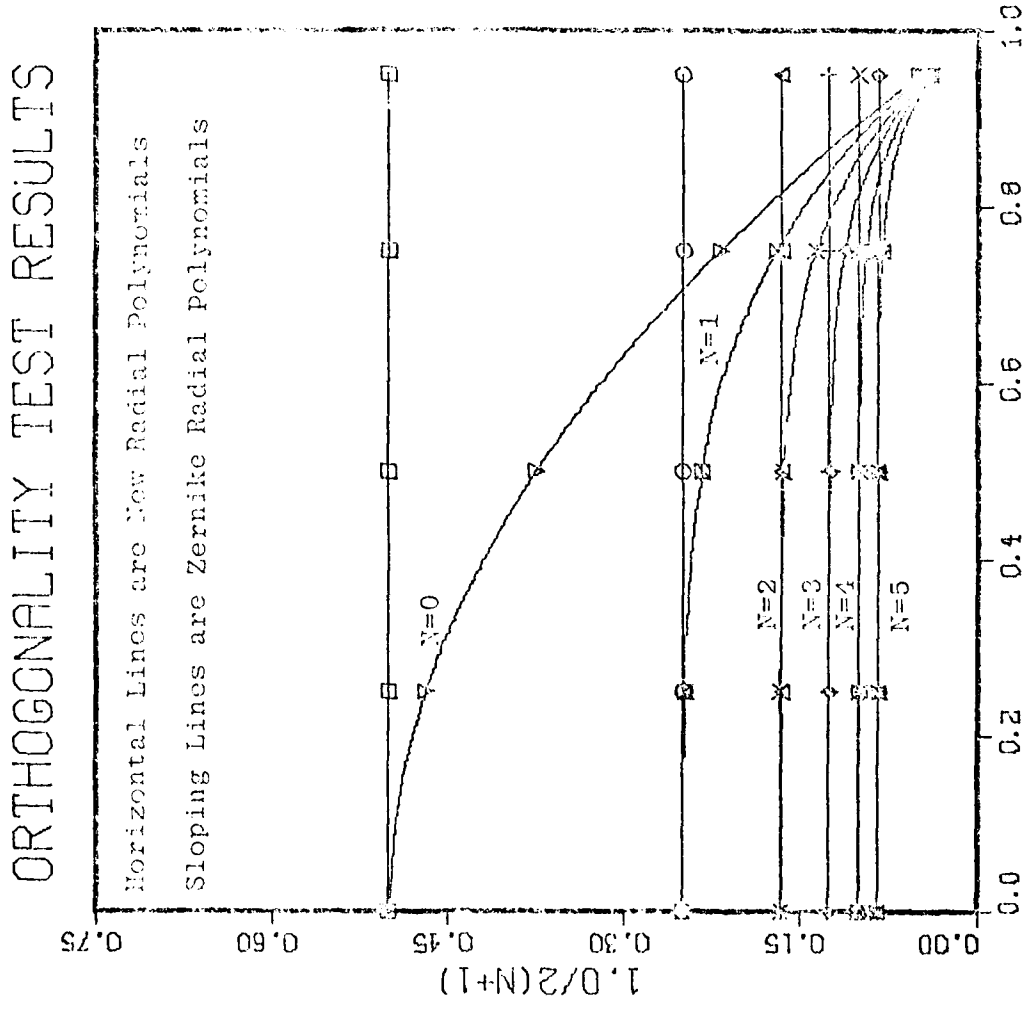


Figure 19. Orthogonality Test Results

With these results, the analysis of wave-fronts can be verified.

Non-annular Verification

To test the software's ability to analyze wave-fronts, two types of wave-fronts were used. One set consisted of generated wave-fronts, and the other ones were actual wave-fronts supplied by AFWL. Both sets were non-annular wave-fronts. By using non-annular wave-fronts, the program's ability to just analyze wave-fronts was tested. With non-annular wave-fronts, the obscuration ratio is zero which results in the two sets of polynomials being identical.

The generated wave-fronts came from a program which had input parameters consisting of the number of coefficients, their values, the center coordinates, and the radius. These wave-fronts verified the operation of the subroutines FAPER, and VALID. These subroutines are the ones which determined the type, center, and radius of the wave-front. With the ability to set the coefficients to specific values, the orthogonality of the polynomials could also be tested. Figure 3 in Chapter II. was generated using this routine.

In every case, FAPER and VALID were able to find the center and radius of the wave-fronts. This was due to the uniformity of the generated wave-fronts. Figure 20 shows the output when the second coefficient of the generated

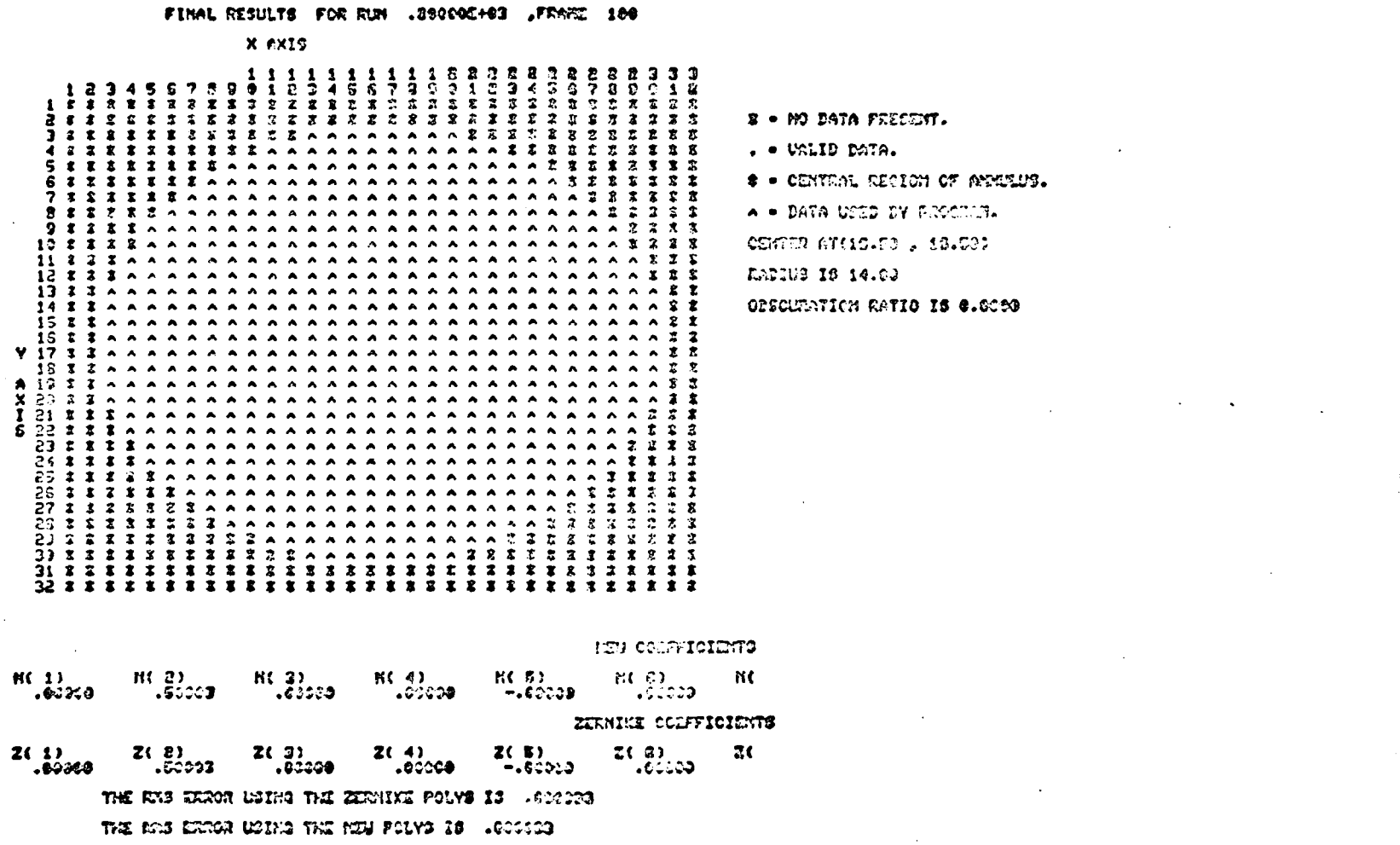


Figure 20. Output of Software with Generated Wave-front

NEW COEFFICIENTS

N(1)	N(2)	N(3)	N(4)	N(5)	N(6)	N(7)	N(8)	N(9)	N(10)	N(11)
.00000	.50003	.00009	.00000	-.00000	.00000	-.00000	-.00030	.00000	.01437	.00000

ZERNIKE COEFFICIENTS

Z(1)	Z(2)	Z(3)	Z(4)	Z(5)	Z(6)	Z(7)	Z(8)	Z(9)	Z(10)	Z(11)
.00000	.50003	.00009	.00000	-.00029	.00000	-.00030	-.00030	.00000	.01437	.00000

THE RMS ERROR USING THE ZERNIKE POLYS IS .014357

THE RMS ERROR USING THE NEW POLYS IS .014357

NEW COEFFICIENTS

N(1)	N(2)	N(3)	N(4)	N(5)	N(6)	N(7)	N(8)	N(9)	N(10)	N(11)
.00000	.50003	.00009	.00000	-.00000	.00000	-.00030	-.00030	.00000	.01437	.00000
N(12)	N(13)	N(14)	N(15)	N(16)	N(17)	N(18)	N(19)	N(20)	N(21)	N(22)
.00000	-.00000	-.00001	-.00000	-.00199	.00000	.01585	-.00000	.01745	-.00000	.00000

ZERNIKE COEFFICIENTS

Z(1)	Z(2)	Z(3)	Z(4)	Z(5)	Z(6)	Z(7)	Z(8)	Z(9)	Z(10)	Z(11)
.00000	.50003	.00009	.00000	-.00029	.00000	-.00030	-.00030	.00000	.01437	.00000
Z(12)	Z(13)	Z(14)	Z(15)	Z(16)	Z(17)	Z(18)	Z(19)	Z(20)	Z(21)	Z(22)
.00000	-.00000	-.00000	-.00000	-.00199	.00000	.01585	-.00000	.01745	-.00000	.00000

THE RMS ERROR USING THE ZERNIKE POLYS IS .027794

THE RMS ERROR USING THE NEW POLYS IS .027794

Figure 21. Output of Software with Generated Wave-front

wave-front was set at 0.5, the radius at 14.0, and the center at 16.5,16.5, Figure 21 is the results of solving for 11 and 22 coefficients. These two figures show the standard output of the program written for this thesis.

As the number of coefficients increased, the RMS error increased from the introduction of extraneous coefficients. These coefficients come about by the approximation of continuous functions over a discrete region. As shown in Figures 20 and 21, the RMS error changes from 0.000033 to 0.027794 waves. This four order of magnitude difference appears to be significant error; however, dividing by the wave length, the RMS error ranges from 0.00031% to 0.26%. Thus even though there is a large change, the deviation from the true wave-front is small. Figures 22-26 show the results of the analysis of a generated wave-front. The RMS error for these plots was 0.026 waves. The difference plots in Figures 25 and 26 are the difference between the actual wave-front and the respective estimated wave-front.

After running over 30 test cases, the RMS error never exceeded 0.050 waves when preset coefficients were within the range of the computed coefficients. For example when only the eleventh coefficient was set at 0.25, the RMS error was 0.24 waves when the first six coefficients were found. The RMS error dropped to 0.024 waves when solving for 11 coefficients. Thus with generated non-annular

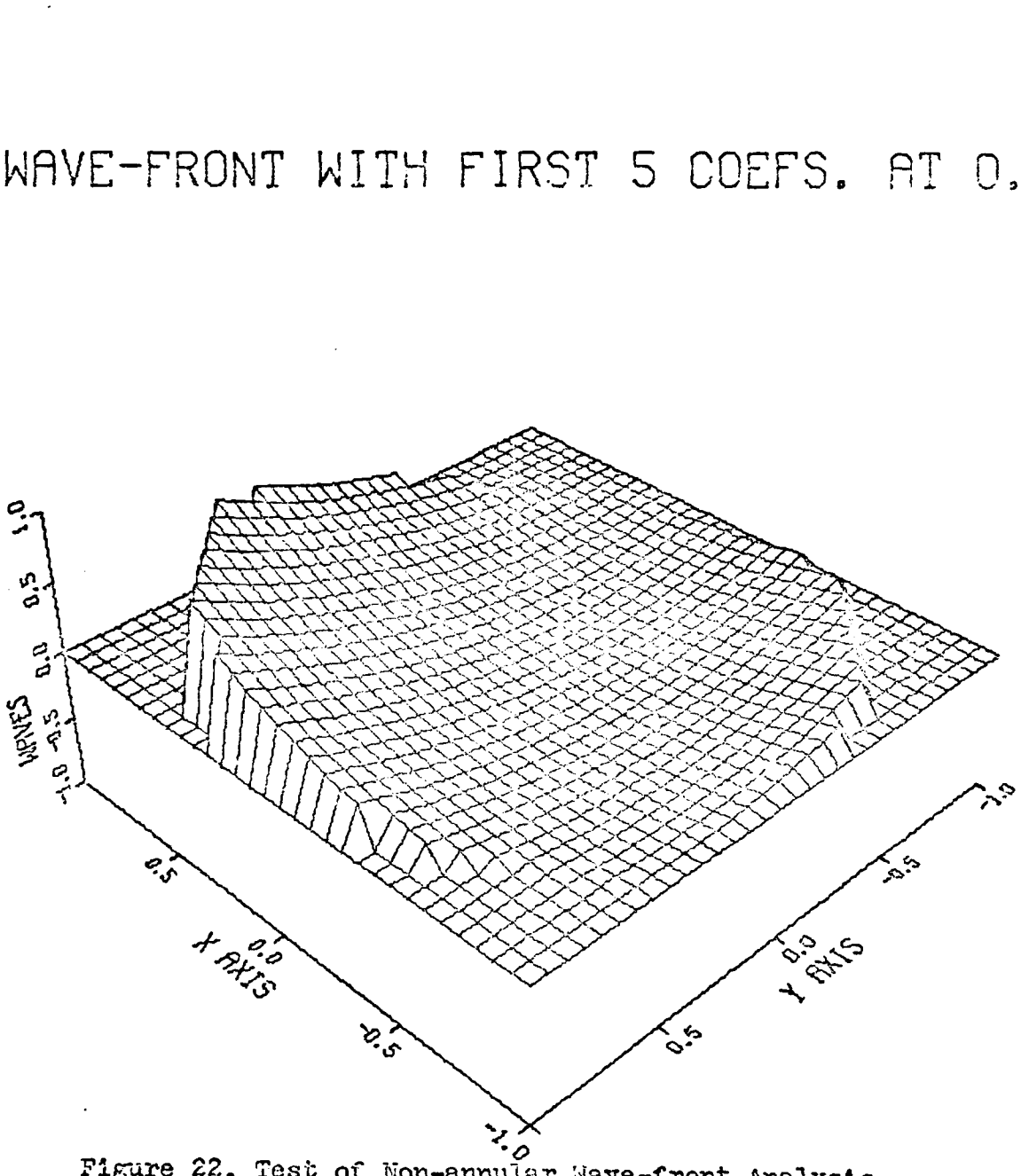


Figure 22. Test of Non-annular Wave-front Analysis

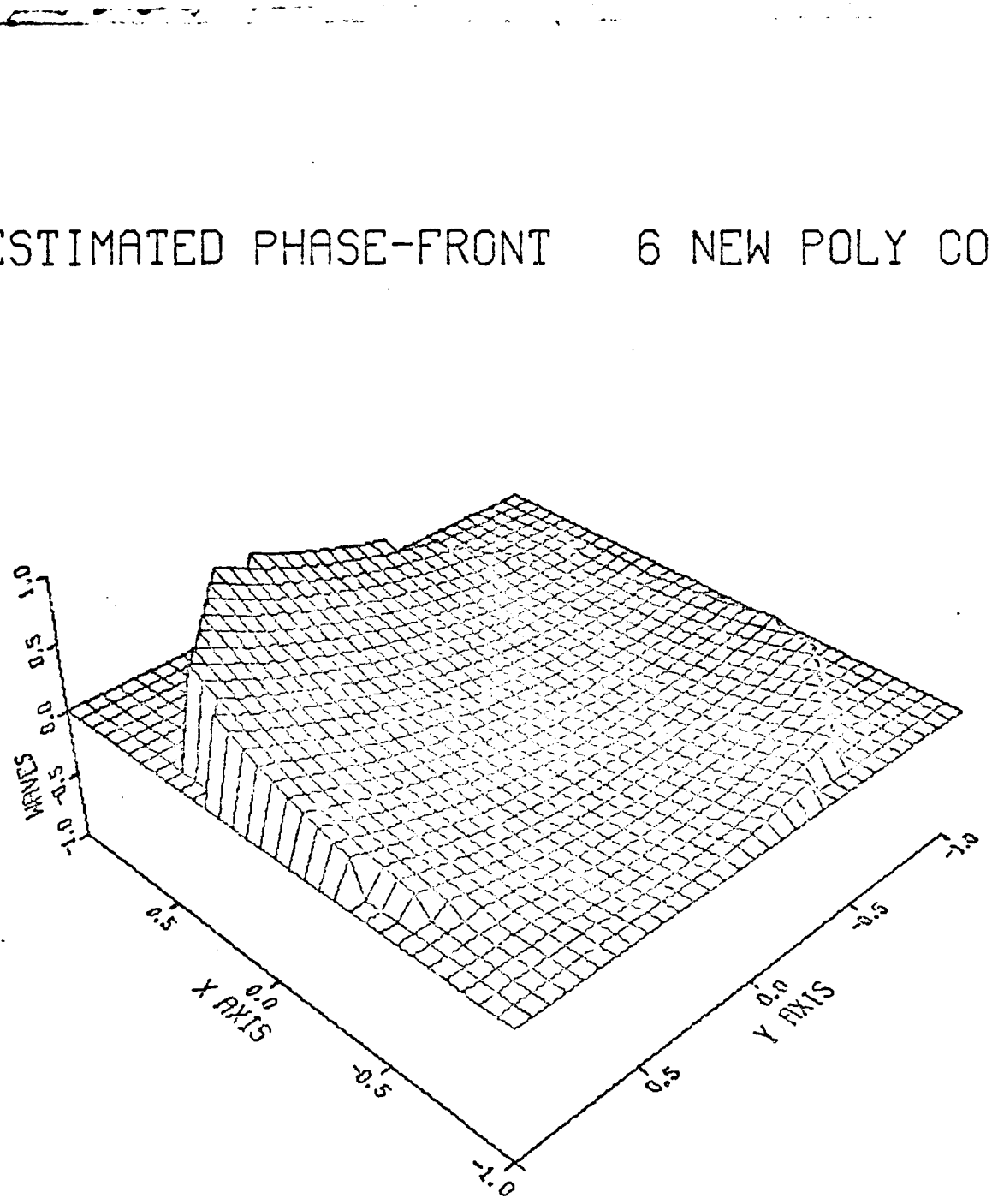


Figure 23. Test of Non-annular Wave-front Analysis

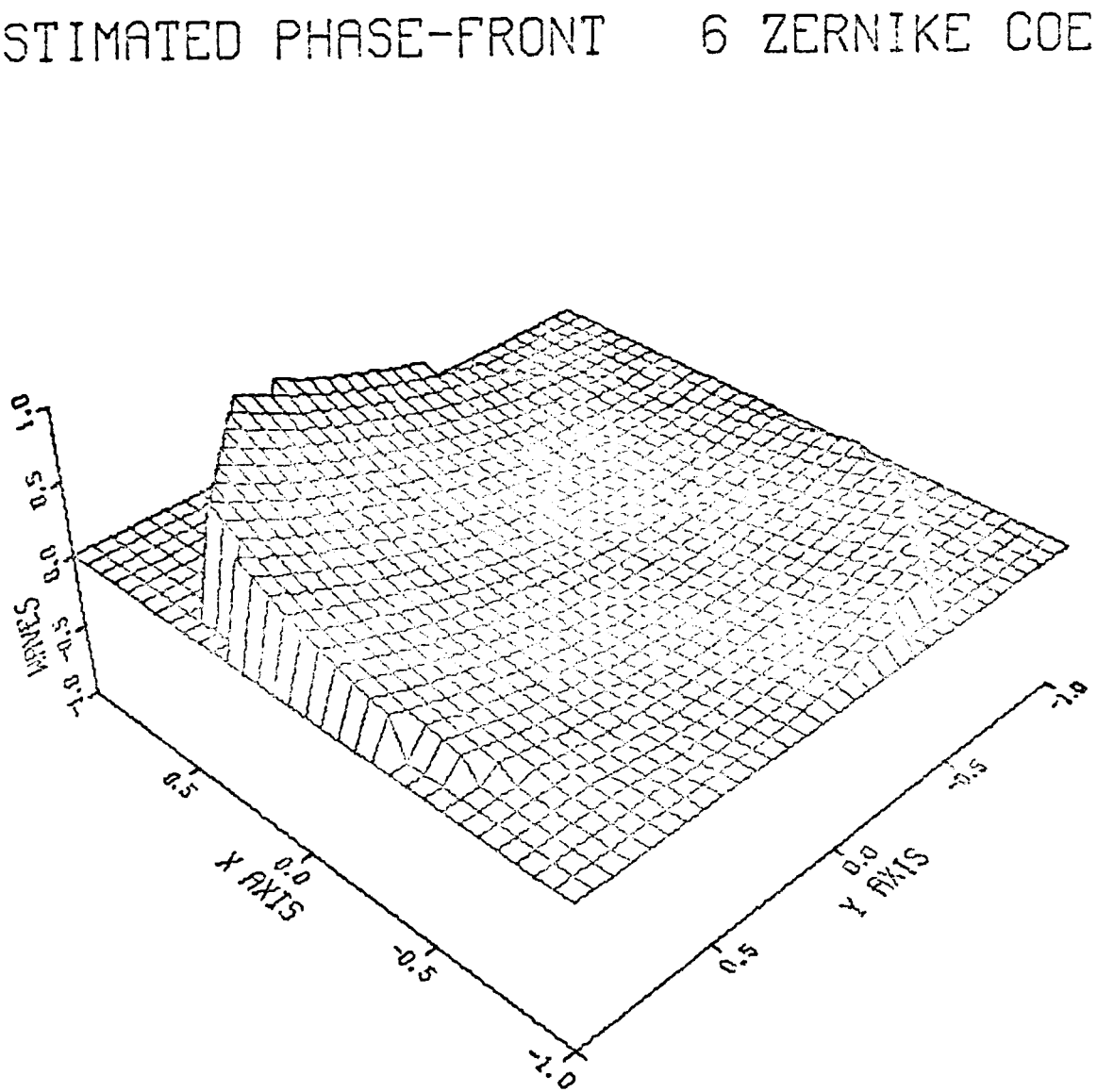


Figure 24. Test of Non-annular Wave-front Analysis

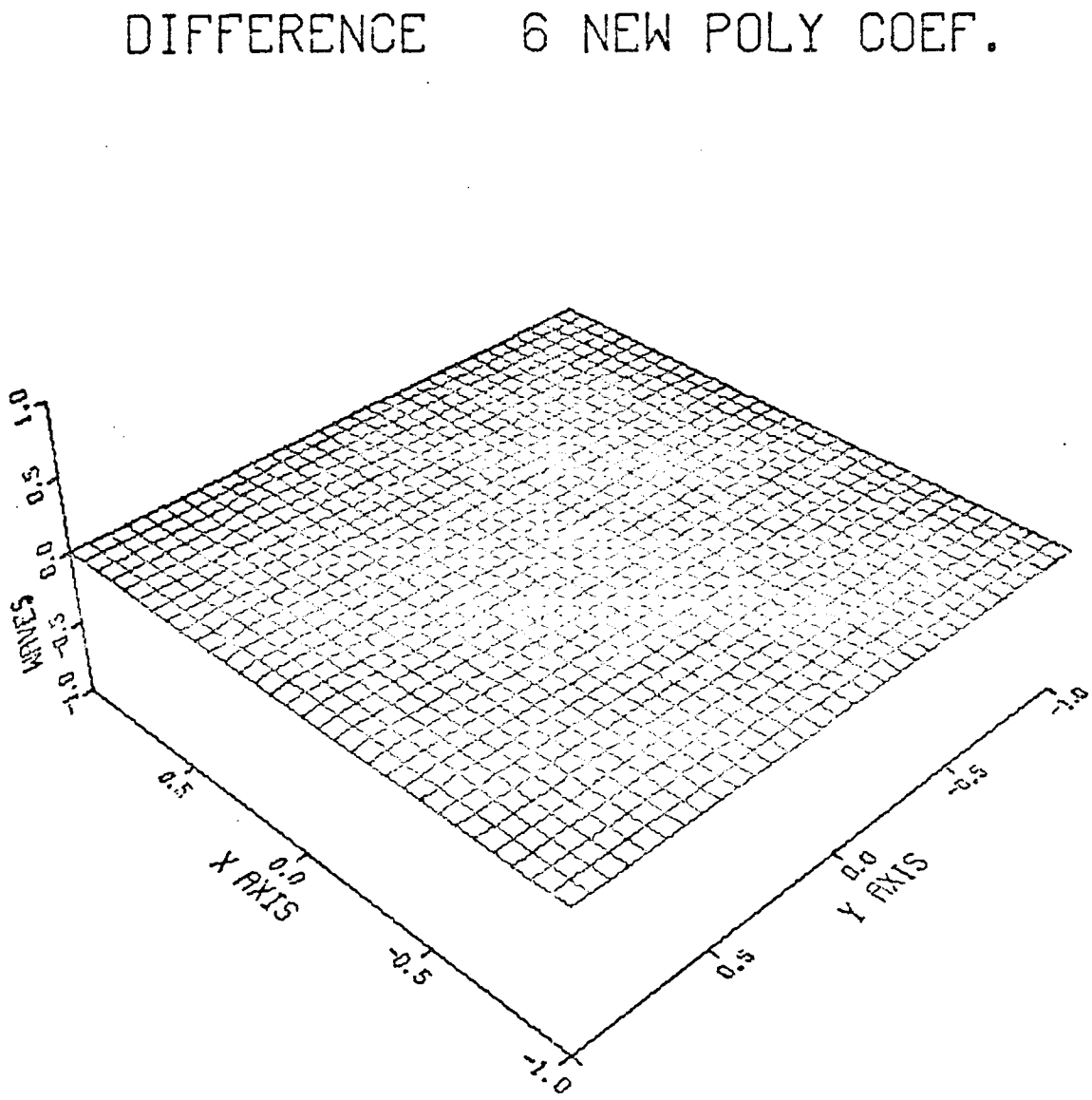


Figure 25. Test of Non-annular Wave-front Analysis

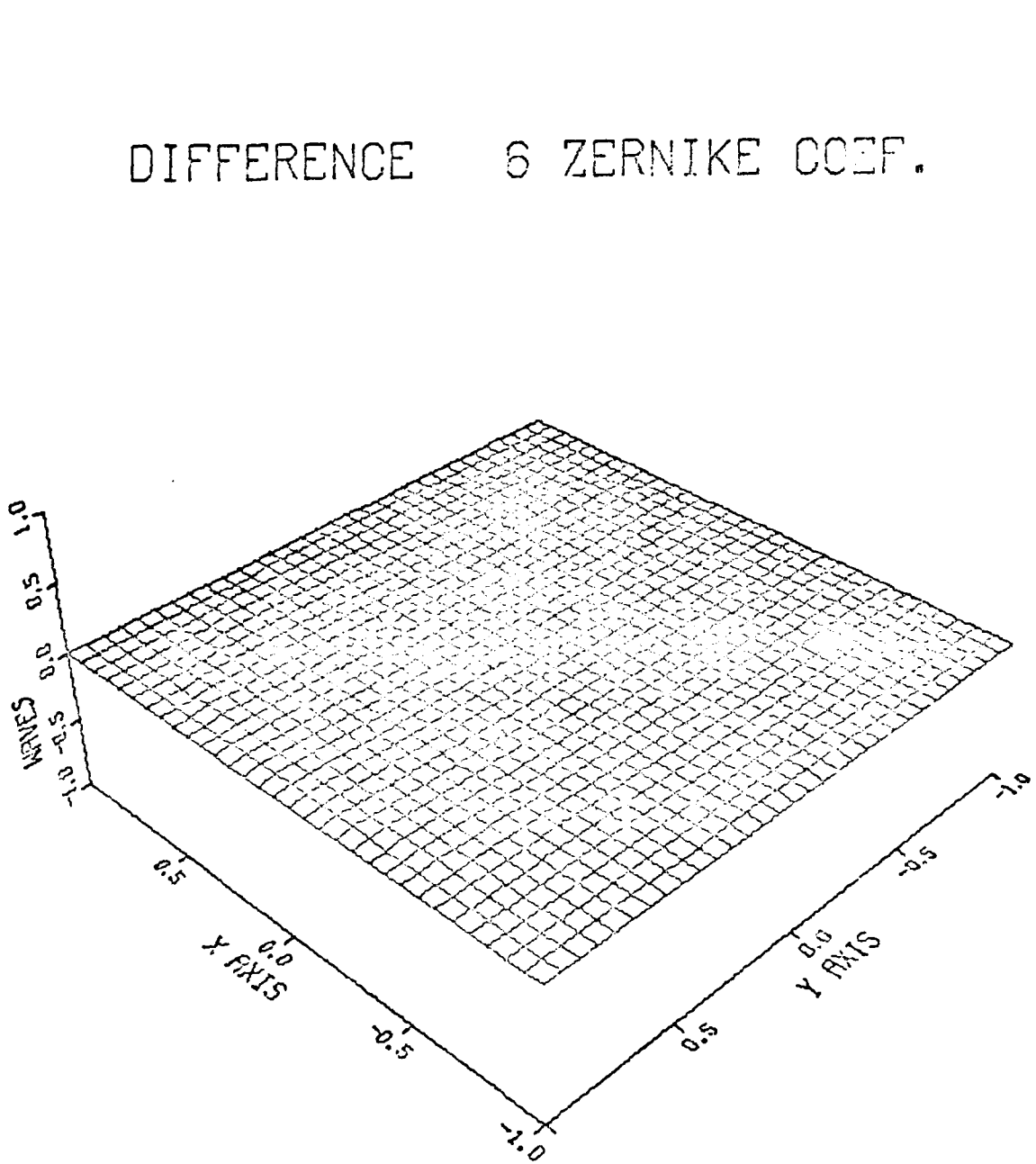


Figure 26. Test of Non-annular Wave-front Analysis

wave-fronts, the software is capable of accurately finding the coefficients with a minimal ammount of error.

The second test of the software with non-annular wave-fronts involved data provided by AFWL. The data consists of 20 sequential frames of wave-front. The first frame is circular and centered in the collection array. As time progressed a knife-edge was inserted in the wave-front blocking off the positive X data points. The knife-edge entered at the top of quadrant I and moved in a -Y direction until the wave-front became a semi-circle at about the twelfth frame of data. The knife-edge was then extracted in the opposite direction until by the twentieth frame the wave-front is nearly circular. The RMS error from six coefficients is plotted in Figure 27. This plot also has the percentage of the wave-front which was obscured by the knife-edge. This figure shows the correspondence between the RMS error and the ammount of wave-front present. In this particular set of data, the wave-front has a large ammount of 45^0 Astigmatism (Zernike polynomial 6). As the +X data was obscured, the coefficient got larger. It was this coefficient which caused the RMS error to increase. Figures 28 to 33 show the frame being analyzed, the estimated wave-front, and the difference between the two. Figures 28 to 30 correspond to frame one in Figure 27 and Figures 31 to 33 correspond to frame twelve in Figure 27. Only the

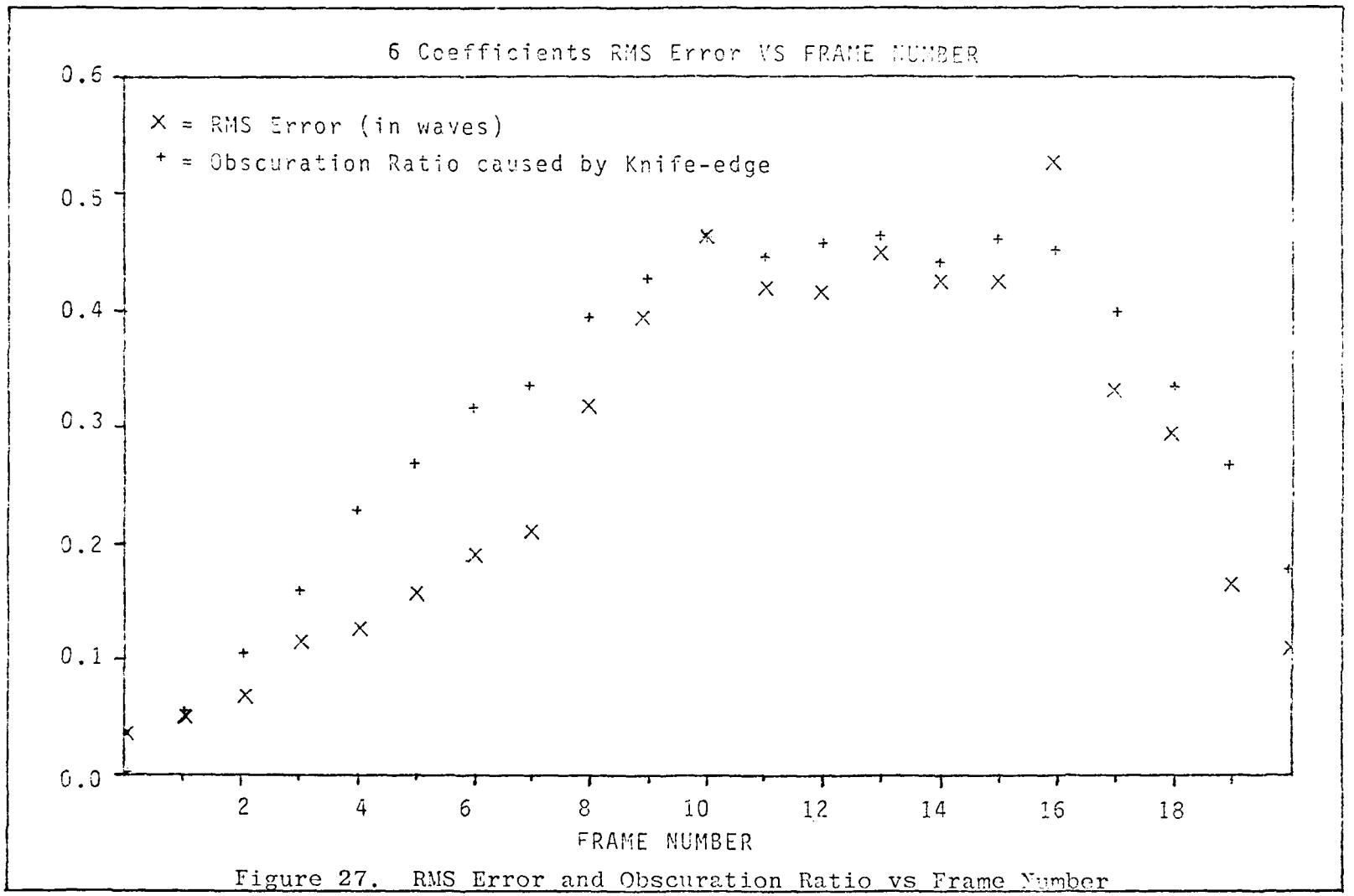
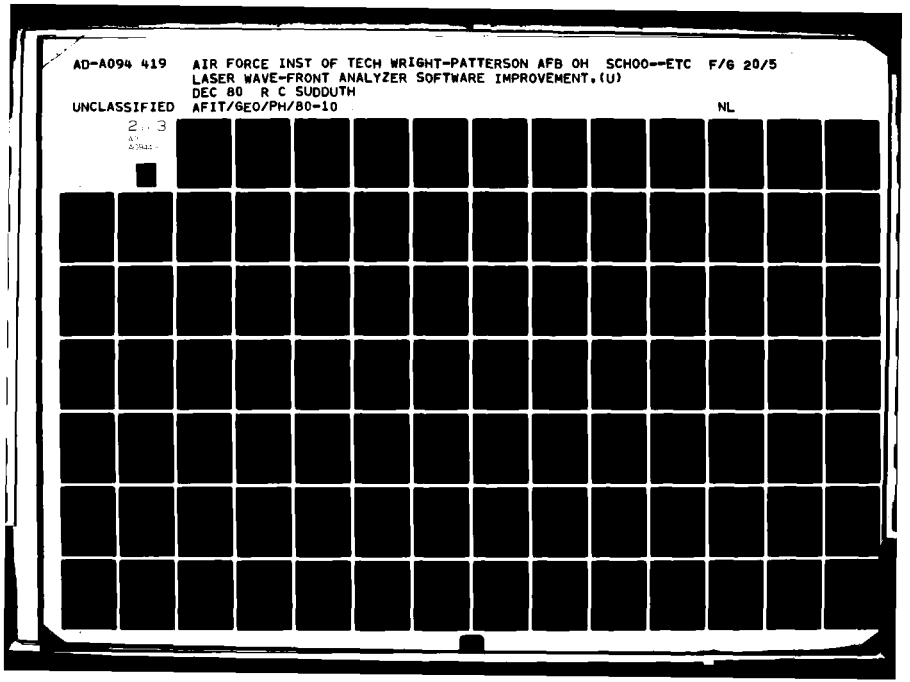


Figure 27. RMS Error and Obscuration Ratio vs Frame Number


AD-A094 419 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 20/5
LASER WAVE-FRONT ANALYZER SOFTWARE IMPROVEMENT,(U)
DEC 80 R C SUDDUTH
UNCLASSIFIED AFIT/GEO/PH/80-10 NL
2 3

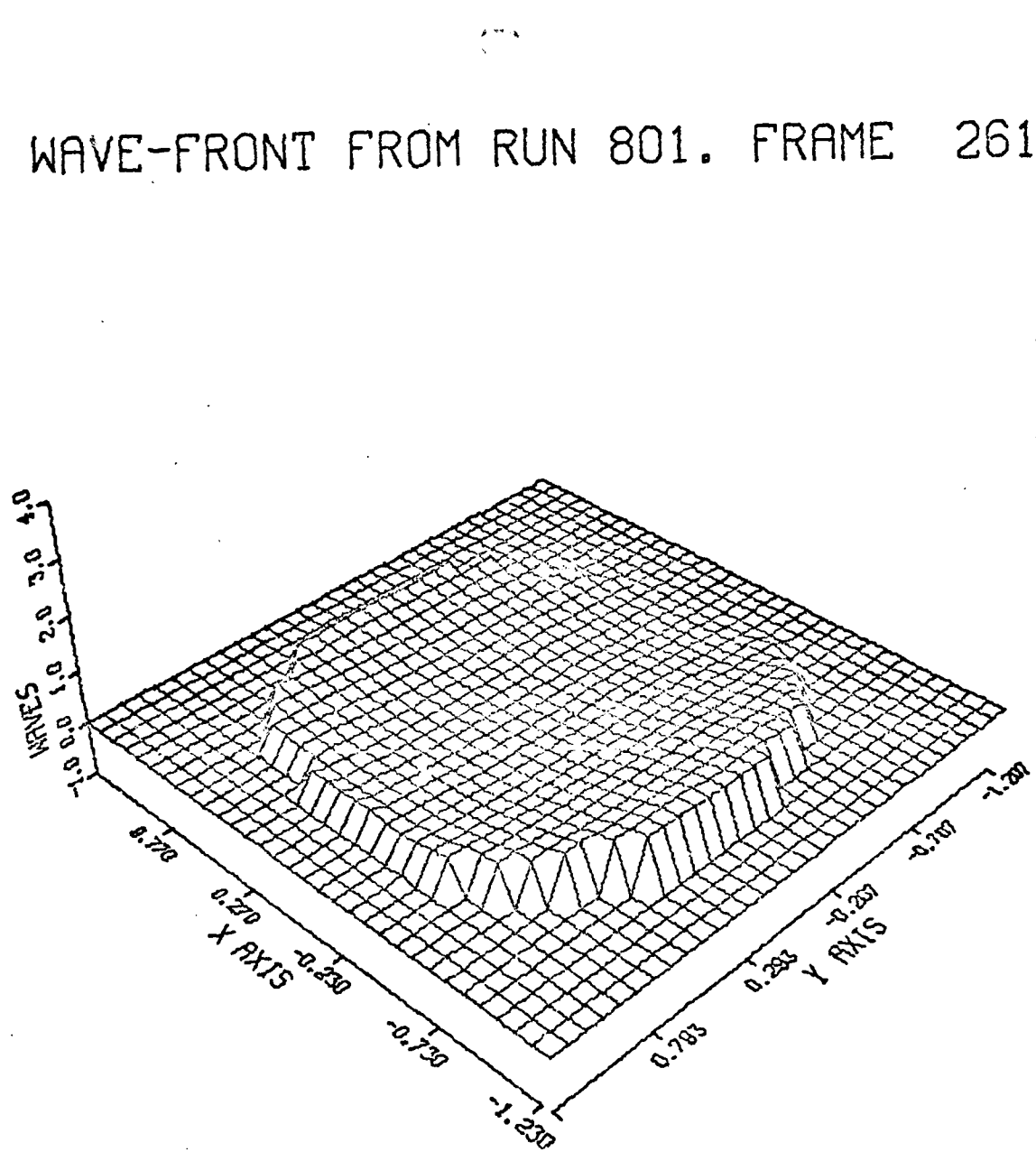


Figure 28. Frame 1, Non-annular Wave-front

ESTIMATED PHASE-FRONT 6 ZERNIKE COEF.

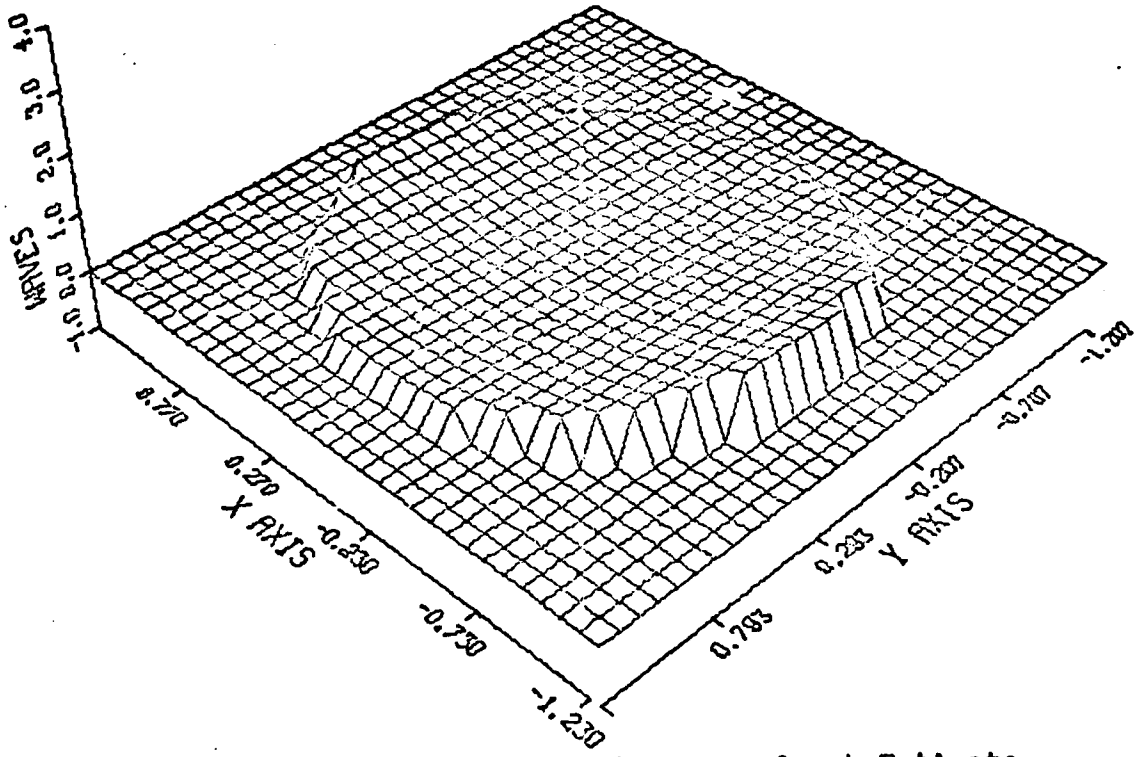


Figure 29. Frame 1, Non-annular Wave-front Estimate

DIFFERENCE 6 ZERNIKE COEF.

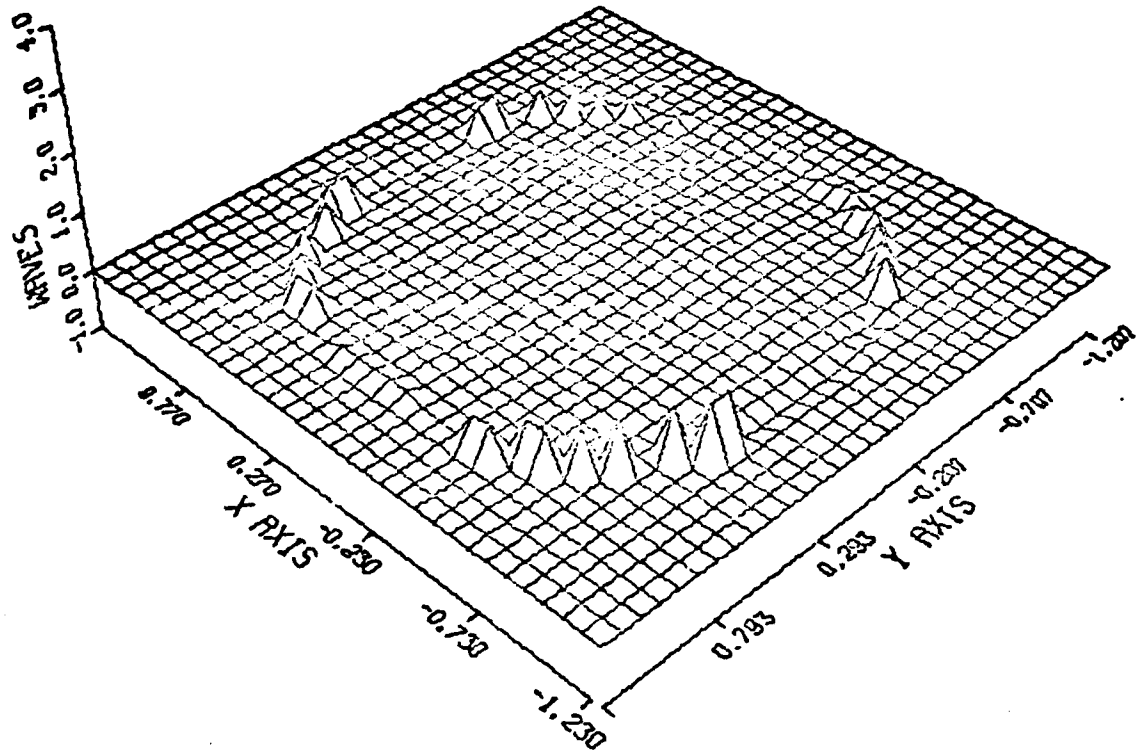


Figure 30. Frame 1, Non-annular Wave-front Difference

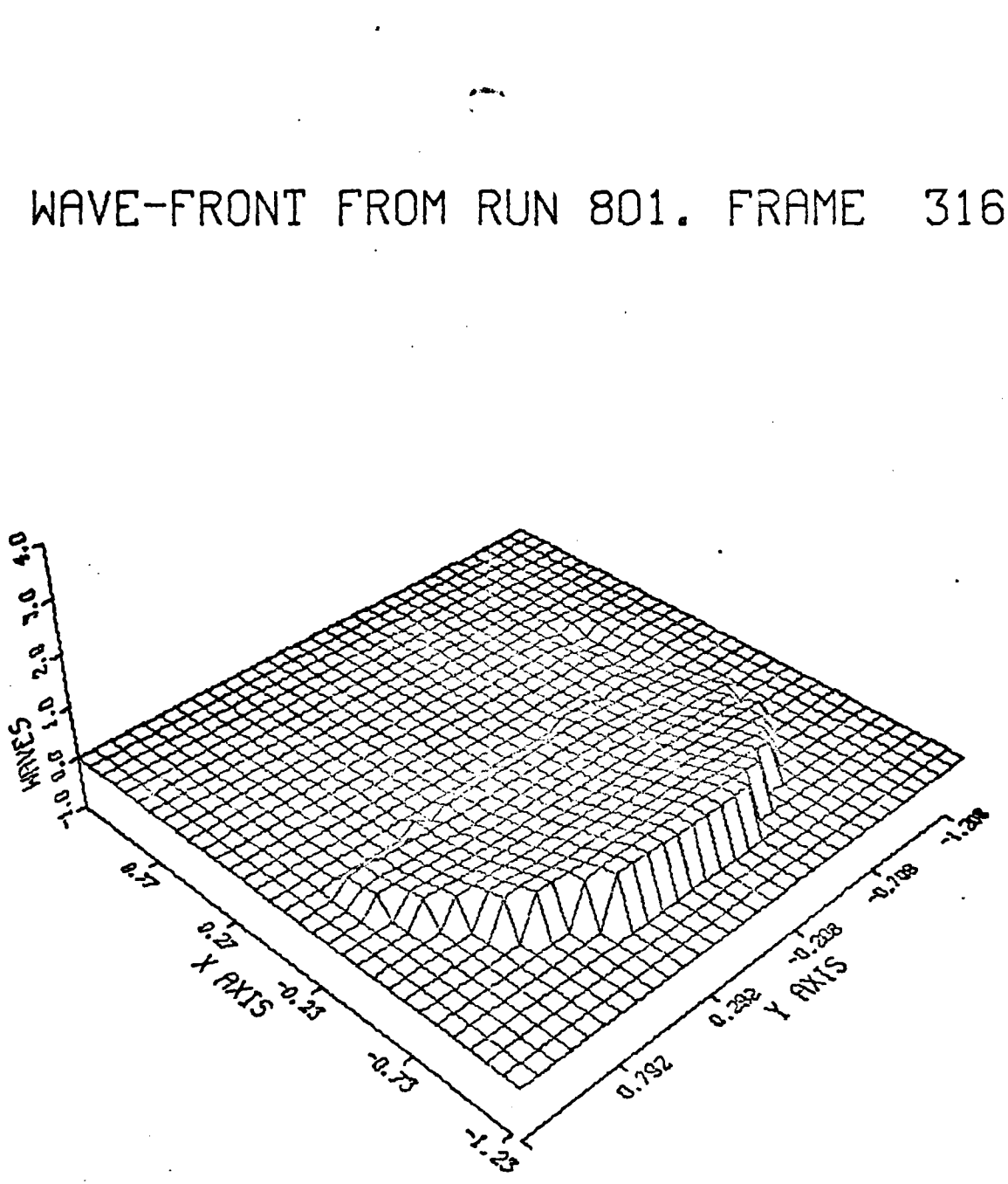


Figure 31. Frame 12, Non-annular Wave-front

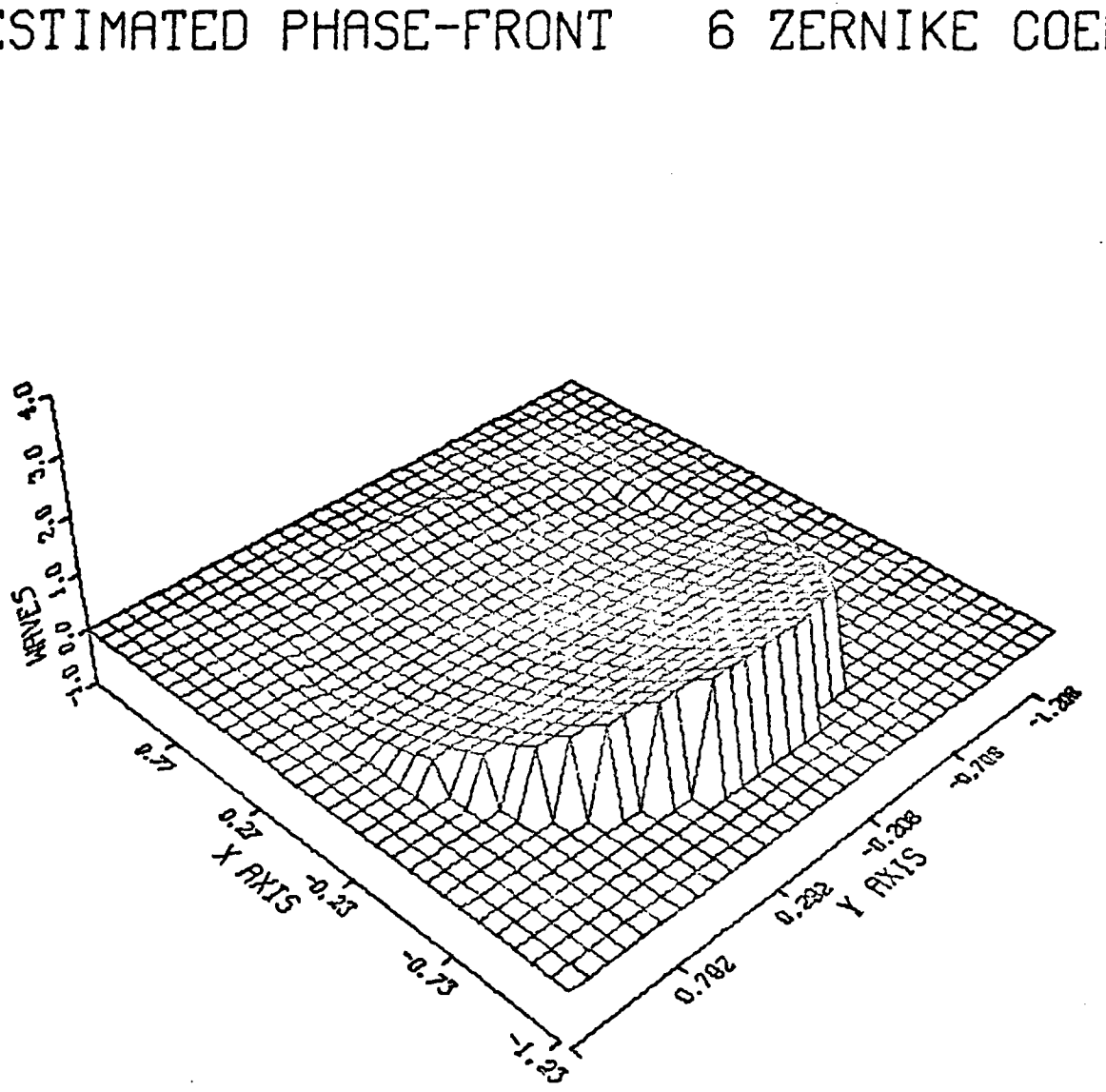


Figure 32. Frame 12, Non-annular Wave-front Estimate

DIFFERENCE 6 ZERNIKE COEF.

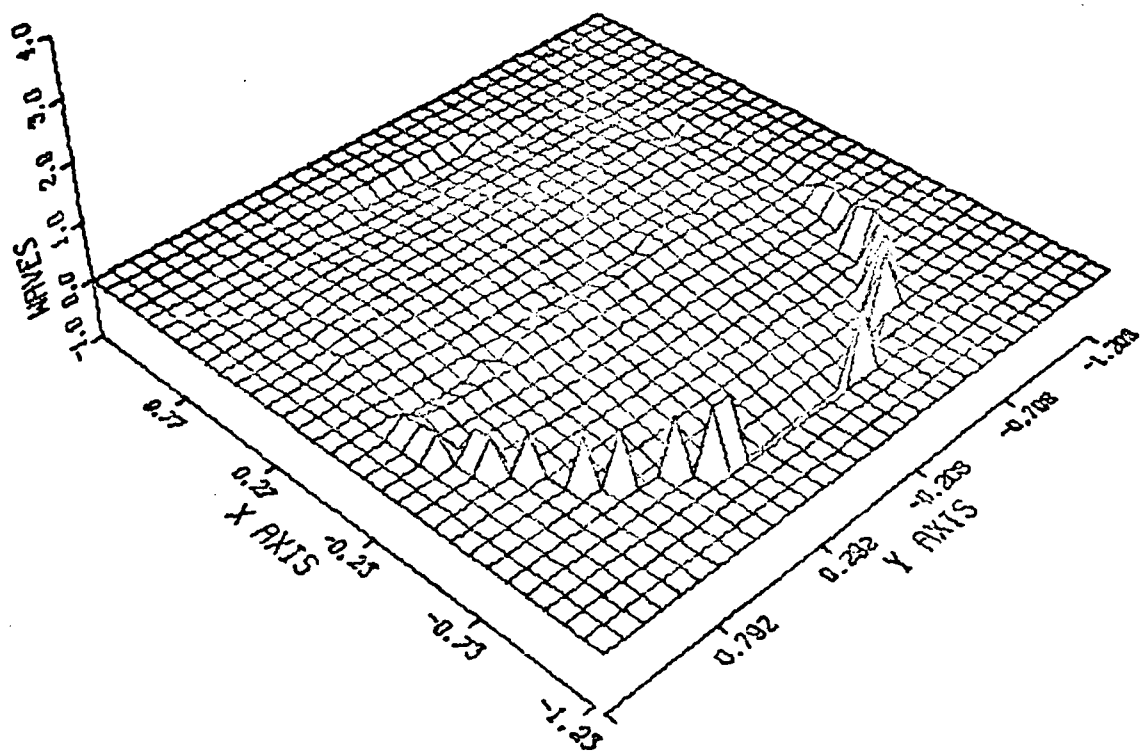


Figure 33. Frame 12, Non-annular Wave-front Difference

Zernike are shown since the two polynomials are the same. The peaks around the perimeter of the difference plots come about by the fitting of circular regions to arbitrary regions. The peaks are not included in the RMS error calculations.

This section has dealt with the analysis of non-annular wave-front. It has been shown that the accuracy of the results is very dependent on the percentage of wave-front the software is able to utilize.

Annular Verification

This section deals with the programs ability to analyze annular wave-fronts. This area will again be split into two types of data: generated and actual. In the case of generated wave-fronts, the same routine is used which generated the non-annular wave-fronts. The only difference is the obscuration ratio is an input parameter; therefore, any size of annular wave-front can be generated. Using these generated wave-fronts, the program was able to find and determine the basic parameters of the wave-front in all cases, even wave-fronts which were not centered. The only time the software made a mistake was when the wave-front filled only one quarter of the array and was not centered. In this position, FAPER found the outside edge first and assumed the wave-front was non-annular. Since the operator can correct problems like this, the parameters were

changed and the program was able to find the coefficients accurately.

The software was able to find the values of the coefficients with the same degree of accuracy as that of non-annular wave-fronts. The closer the obscuration ratio was to the actual value, the closer the RMS error of the two sets of polynomials. For example, when a non-annular wave-front was analyzed as an annular one, the RMS error of the new polynomials were about twice that of the Zernike polynomials. As the obscuration ratio decreased, the RMS error of the new polynomials would approach the RMS error of the Zernike polynomials. Thus the best results were obtained when the actual and estimated obscuration ratios matched. The same results were seen when the input wave-front was annular. The RMS error of the new polynomials; however, when the obscuration ratio exceeded about 0.6 this was not always the case. In this regime the program had difficulty computing the coefficients of the polynomials. Just as the RMS error increased in the non-annular wave-front when it was obscured with a knife-edge, the same happened with the RMS error as the obscuration ratio increased above 0.6.

It must be explained that the Zernike coefficients are not obtained from analyzing the wave-front, but from converting the new polynomial coefficients to Zernike coefficients. If the Zernike coefficients were used, the

wave-front would always have to be assumed to be non-annular. If the Zernike polynomials were used on an annular wave-front, their lack of orthogonality would result in severe cross-coupling. Therefore even though either set may give a slightly better result in the RMS error, the main point is the wave-front was analyzed with an orthogonal set of polynomials.

When actual annular wave-fronts from AFWL were analyzed, the biggest problem was entering the proper parameters. The wave-front used was off-center and part of the beam was obscured. Figure 34 shows the annular wave-front. This was very difficult to get good results from, since the software would only use a small portion of the wave-front. The best results were obtained when the outside radius was increased to include all of the points of the array. When the radius was increased from 9.25 to 17.0 and the obscuration ratio was changed accordingly, the RMS error dropped from 2.88 to 1.07 waves. This error was further reduced when the center was moved to the approximate center of the data, and the obscuration ratio was dropped from 0.67 to 0.15 waves. This change resulted in the RMS error dropping to 0.31 waves or 2.92%.

Thus the software is able to analyze annular wave-fronts as designed. It is able to find the coefficients of the new polynomials and convert them to the corresponding Zernike coefficients. The next chapter presents the recommendations and conclusions of this thesis.

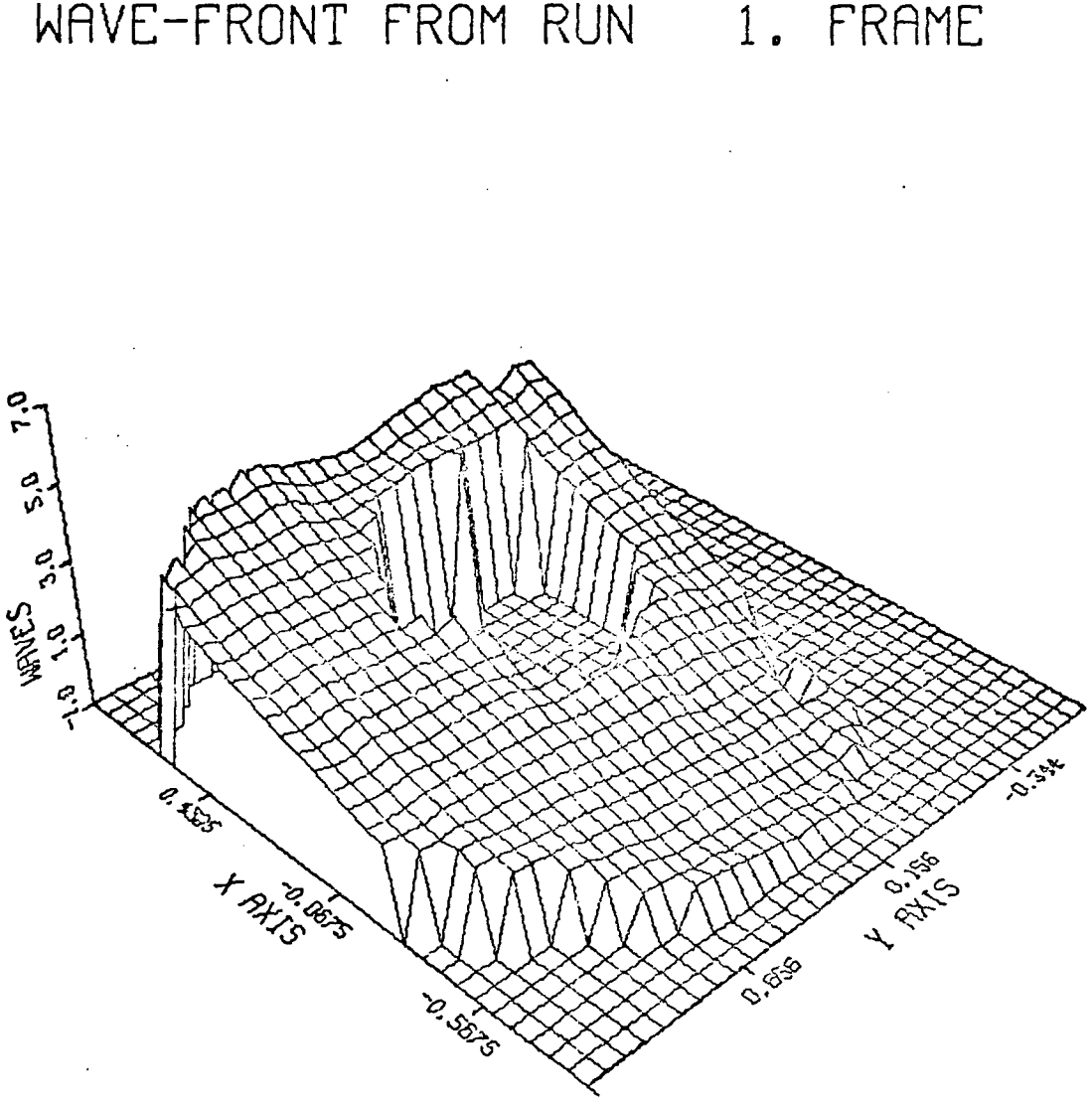


Figure 34. AFWL's Annular Wave-front

VI. Conclusions and Recommendations

This thesis has presented a method of analyzing annular wave-fronts. Starting with a brief description of the problem in Chapter I, the second chapter presented some insight into how a LWA works. Next the Patterson and Zernike polynomials were compared and their respective faults were presented. Chapter III gave the reader the theory behind the construction of an orthogonal set of polynomials, finishing with a means of converting Zernike polynomials to a new set of orthogonal polynomials and vice versa. Chapter IV presented the development of the software necessary to analyze annular wave-fronts, and Chapter V verified the software's operation.

In summary, the LWA software package developed here is able to find the coefficients of up to and including the first 22 Zernike polynomials. The wave-front being analyzed can range in shape from circular to an annulus with an obscuration ratio of over 0.95. The best results were obtained when the obscuration ratio did not exceed 0.60. The software performed best when the true center, obscuration ratio, and outside radius were known. With these parameters coupled with the total wave-front area present, the resulting RMS error was consistently less than 0.040 waves.

This study has met all of its goals which were: (1) using an orthogonal set of polynomials, find the coefficients describing the wave-front, (2) these coefficients must not change as the number of coefficients changes, (3) find the coefficients as fast as possible. On a CDC 6600 the software was, on the average, able to find 6, then 11, then 22 coefficients of one frame of data in 20 seconds of CPU time. With these results in mind, the next section considers possible improvements to the software.

Recommendations

The current software assumes the wave-front is either circular or annular. One way to improve the software would be to have the software define the region and then generate a set of polynomials which are orthogonal over the defined region. This would allow the software to use all of the valid data points. This could be further modified by weighting the phase data values by the intensity at each point. In so doing, those phase values whose corresponding intensity is very small would have less importance than those with large intensity; thereby giving less emphasis to phase data the system was just able to detect.

Bibliography

1. Born, M. and E. Wolf. Principles of Optics (Second edition). Oxford: Pergamon Press Limited, 1964.

2. Electro-Optical Division. Laser Wavefront Analyzer Computer Software Technical Description. Report 12314-1. Norwalk, Connecticut: Perkin-Elmer Corporation, May 1977.

3. Noble, B. and J.W. Daniel. Applied Linear Algebra. Englewood Cliffs, N.J.: Prentice-Hall, Inc.,1977.

4. Patterson, J.S. "Interferometric Determination of Optical Performance," Engineering Report 129,Optical Technology Division. Norwalk, Connecticut: Perkin-Elmer Corporation, August 1969.

5. Rimmer, M.P. "Method for Evaluating Lateral Sheering Interferograms," Applied Optics,13:623-629 (1974).

6. Tatian, B. "Aberration Balancing in Rotationally Symmetric Lenses," Journal of the Optical Society of America,64(8):1083:1091 (August 1974).

7. Van Workum, J., J.A. Plascyk, and M.L. Skolnick. "Laser Wavefront Analyzer for Diagnosing High-Energy Lasers," SPIE Adaptive Optical Components,141:58-66 (1978).

8. Wang, J.Y. and D.E. Silva. "Wave-front Interpretation with Zernike Polynomials," Applied Optics, 19(9):1510-1518 (May 1980).

Appendix A

Wave-front Construction from Delta Phase Data

Appendix A

Wave-front Construction from Delta Phase Data

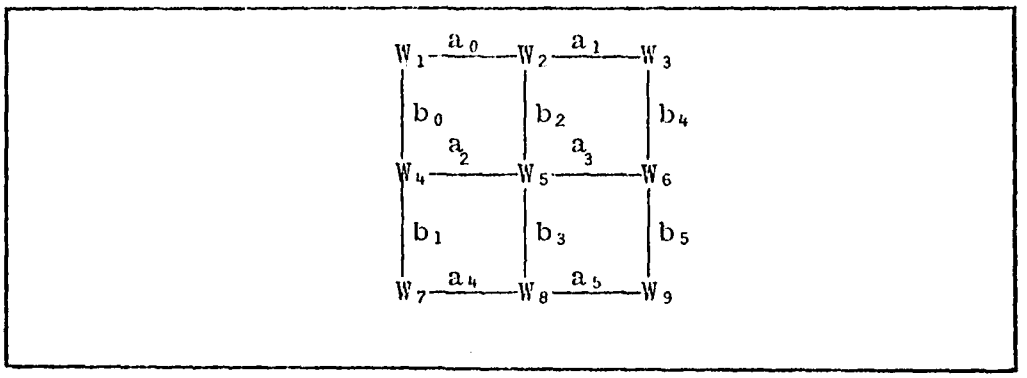


Figure 35. Simple Wave-front Collection Array

The process of finding the wave-front from the delta phase measurements involves using the least-squares technique. Figure 35 shows a simple collection array made up of nine points. The a_i's and b_i's are the X and Y delta phase measurements respectively. The least-squares method of finding the wave-front involves minimizing the square of the difference between the estimated phase value at each point (W_i) and the measured phase difference (a_i or b_i). For the simple array in Figure 35, the following equations result:

$$\sigma_1 = (W_2-W_1-a_0)^2 + (W_4-W_1-b_0)^2 \qquad (A.1)$$

$$\sigma_2 = (W_2-W_1-a_0)^2 + (W_3-W_2-a_1)^2 + (W_5-W_2-b_2)^2 \quad (A.2)$$

$$\sigma_3 = (W_3-W_2-a_1)^2 + (W_6-W_3-b_4)^2 \quad (A.3)$$

$$\sigma_4 = (W_4-W_1-b_0)^2 + (W_7-W_4-b_1)^2 + (W_5-W_4-a_2)^2 \quad (A.4)$$

$$\sigma_5 = (W_6-W_5-a_3)^2 + (W_5-W_4-a_2)^2 + (W_5-W_2-b_2)^2 + (W_8-W_5-b_3) \quad (A.5)$$

$$\sigma_6 = (W_6-W_3-b_4)^2 + (W_6-W_5-a_3)^2 + (W_9-W_6-b_5)^2 \quad (A.6)$$

$$\sigma_7 = (W_7-W_4-b_1)^2 + (W_8-W_7-a_4)^2 \quad (A.7)$$

$$\sigma_8 = (W_8-W_7-a_4) + (W_9-W_7-a_5) + (W_8-W_5-b_3) \quad (A.8)$$

$$\sigma_9 = (W_9-W_8-a_5) + (W_9-W_6-b_5) \quad (A.9)$$

Thus to minimize the sum of the square of the differences, the derivative of Equations (A.1-9) is with respect to the phase value at each point, or $\partial\sigma_i/\partial W_i$. Each derivative is set equal to zero. Doing this to Equations (A.1-9) and simplifying yields:

$$2W_1-W_2-W_4 = -a_0-b_0 \quad (A.10)$$

$$3W_2-W_1-W_3-W_5 = a_0-a_1-b_2 \quad (A.11)$$

$$2W_3 - W_2 - W_6 = a_1 - b_4 \qquad \text{(A.12)}$$

$$3W_4 - W_1 - W_5 - W_7 = b_0 - b_1 - a_2 \qquad \text{(A.13)}$$

$$4W_5 - W_2 - W_4 - W_6 - W_8 = a_2 - a_3 + b_2 - b_3 \qquad \text{(A.14)}$$

$$3W_6 - W_3 - W_5 - W_9 = a_3 + b_4 - b_5 \qquad \text{(A.15)}$$

$$2W_7 - W_4 - W_8 = b_1 - a_4 \qquad \text{(A.16)}$$

$$3W_8 - W_5 - W_7 - W_9 = a_4 + b_3 - a_5 \qquad \text{(A.17)}$$

$$2W_9 - W_6 - W_8 = a_5 + b_5 \qquad \text{(A.18)}$$

Putting Equations (A.10-18) into matrix form yields

$$\begin{bmatrix} 2 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 3 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 3 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \\ W_5 \\ W_6 \\ W_7 \\ W_8 \\ W_9 \end{bmatrix} = \begin{bmatrix} -a_0 - b_0 \\ a_0 - a_1 - b_2 \\ a_1 - b_4 \\ b_0 - b_1 - a_2 \\ a_2 - a_3 + b_2 - b_3 \\ a_3 + b_4 - b_5 \\ b_1 - a_4 \\ a_4 + b_3 - a_5 \\ a_5 + b_5 \end{bmatrix}$$

This matrix can be expressed as $\underline{A}\underline{W} = \underline{B}$. To find the values of the phase at each point, the matrix $\underline{A}$ is inverted and multiplied times $\underline{B}$. The matrix $\underline{A}$ is Hermitian, irreducibly diagonally dominant, and all of the

diagonal entries are positive real numbers (Ref.5) regardless of the order of the array. Usually one of the points in the phase array is set to zero as a starting point. Following the procedure outlined above, the LWA performs the same operation on a 32 by 32 collection array to construct the wave-front used by the software in this thesis.

Appendix B

Plots of the First 22

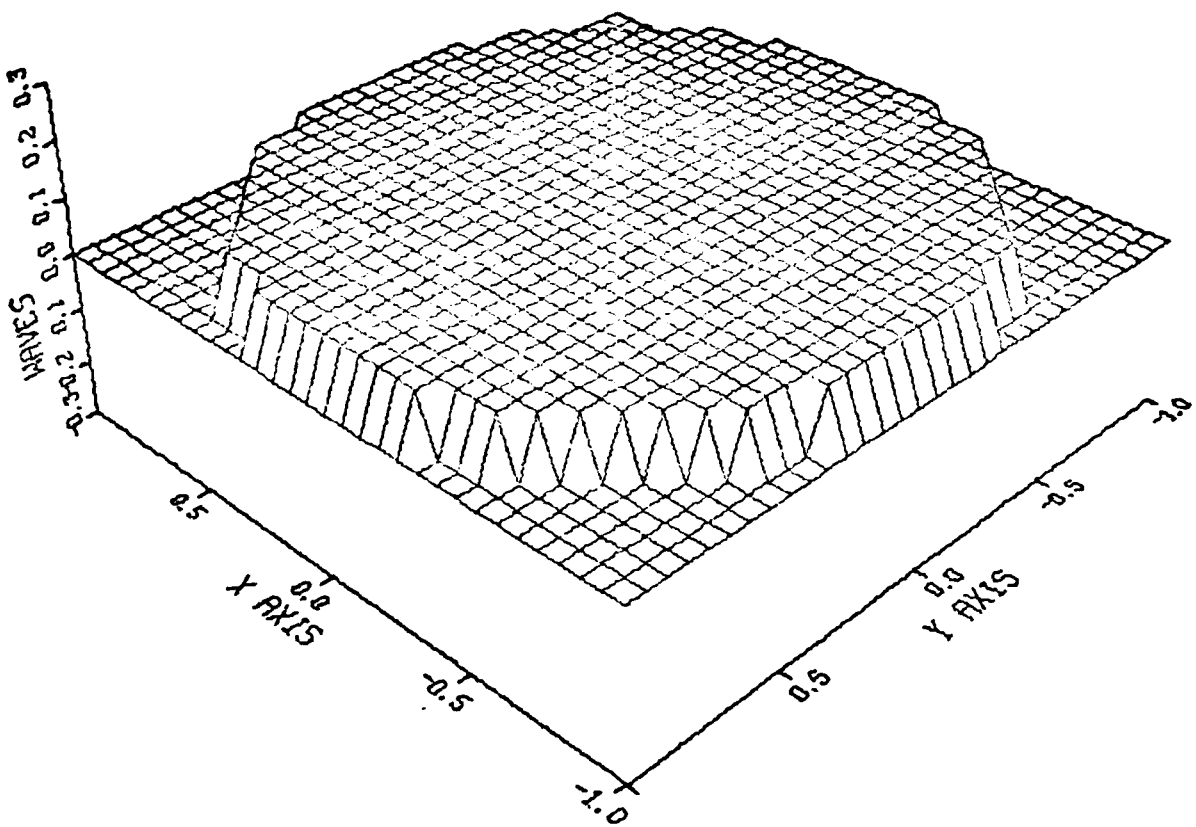
Zernike Polynomials

Appendix B

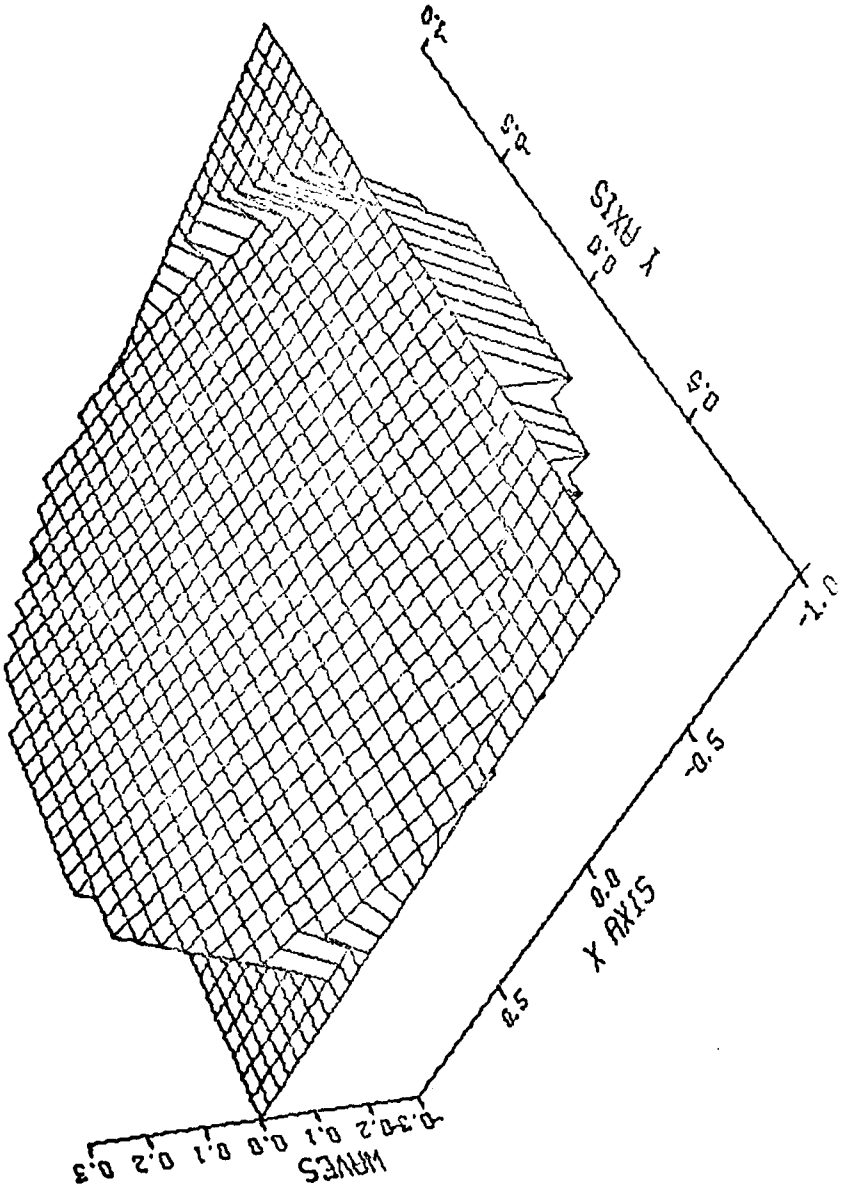
Plots of the First 22 Zernike Polynomials

The next 22 pages show the reader what the first 22 terms of the Zernike polynomials look like individually. As pointed out in Table II, the first ten polynomials are familiar to those in the field of optics. Each plot is the result of setting the corresponding coefficient to a value of 0.1 wave.

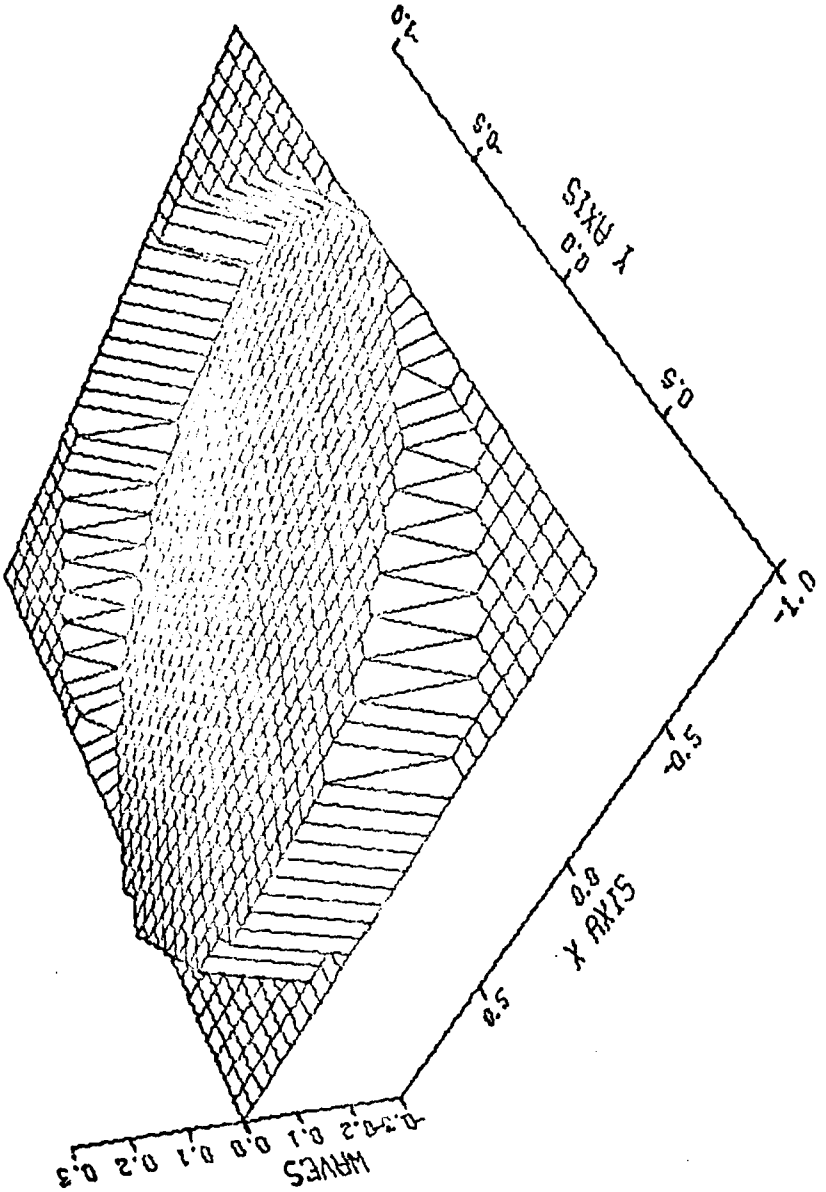
ZERNIKE COEF 1, SET AT 0.1



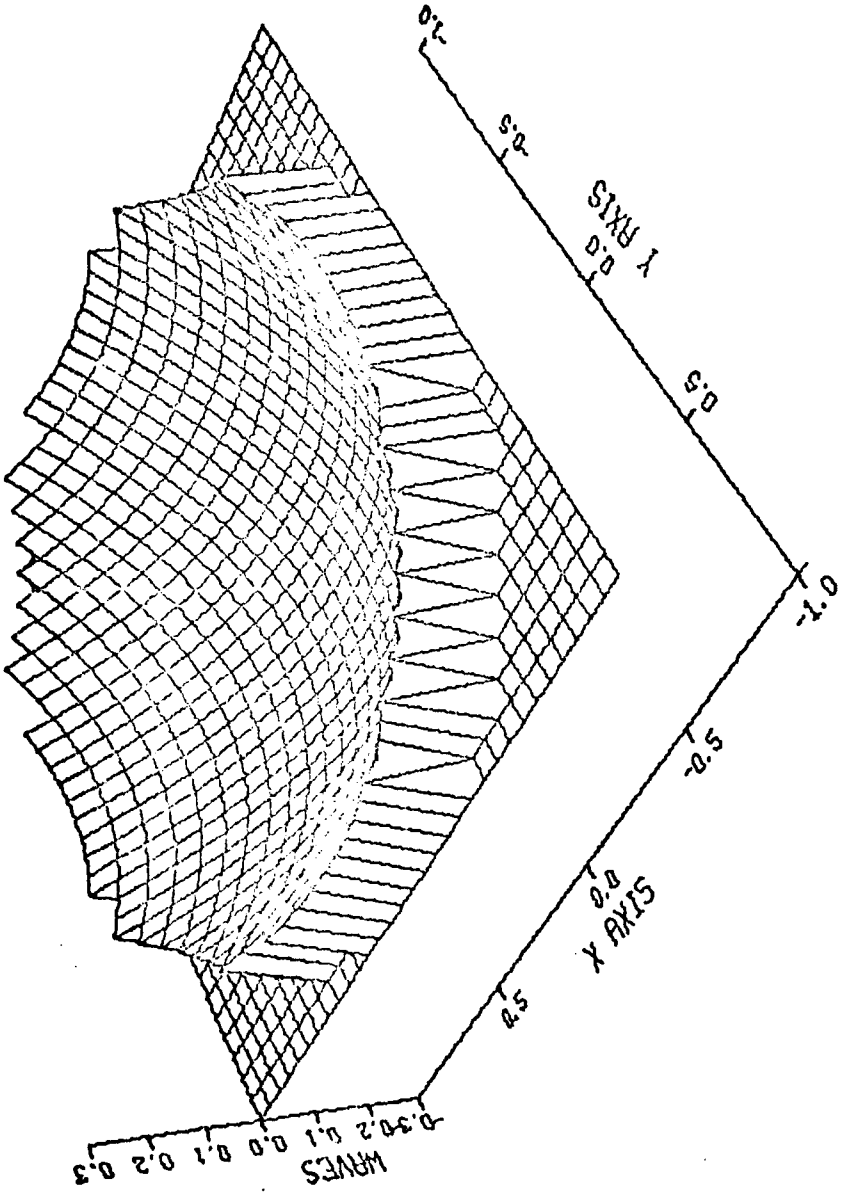
ZERNIKE COEF 2, SET AT 0.1



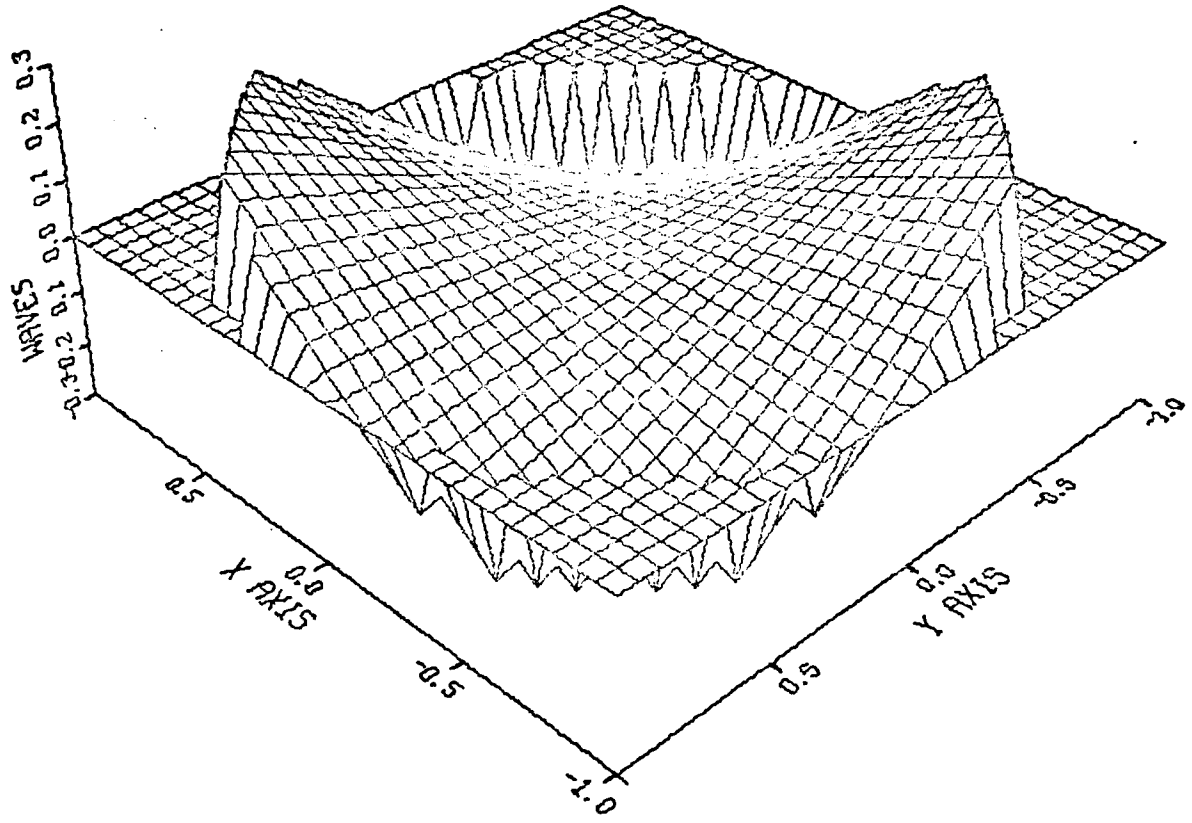
ZERNIKE COEF 3, SET AT 0.1



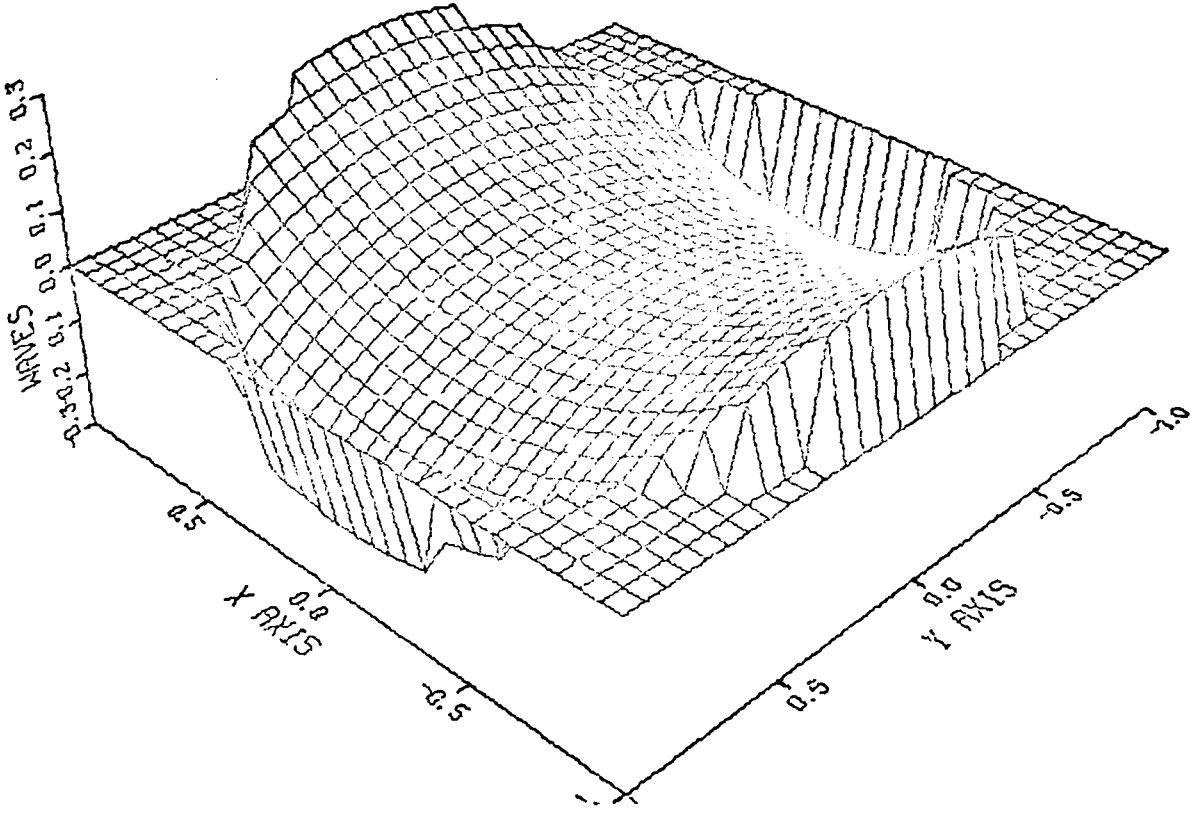
ZERNIKE COEF 4, SET AT 0.1



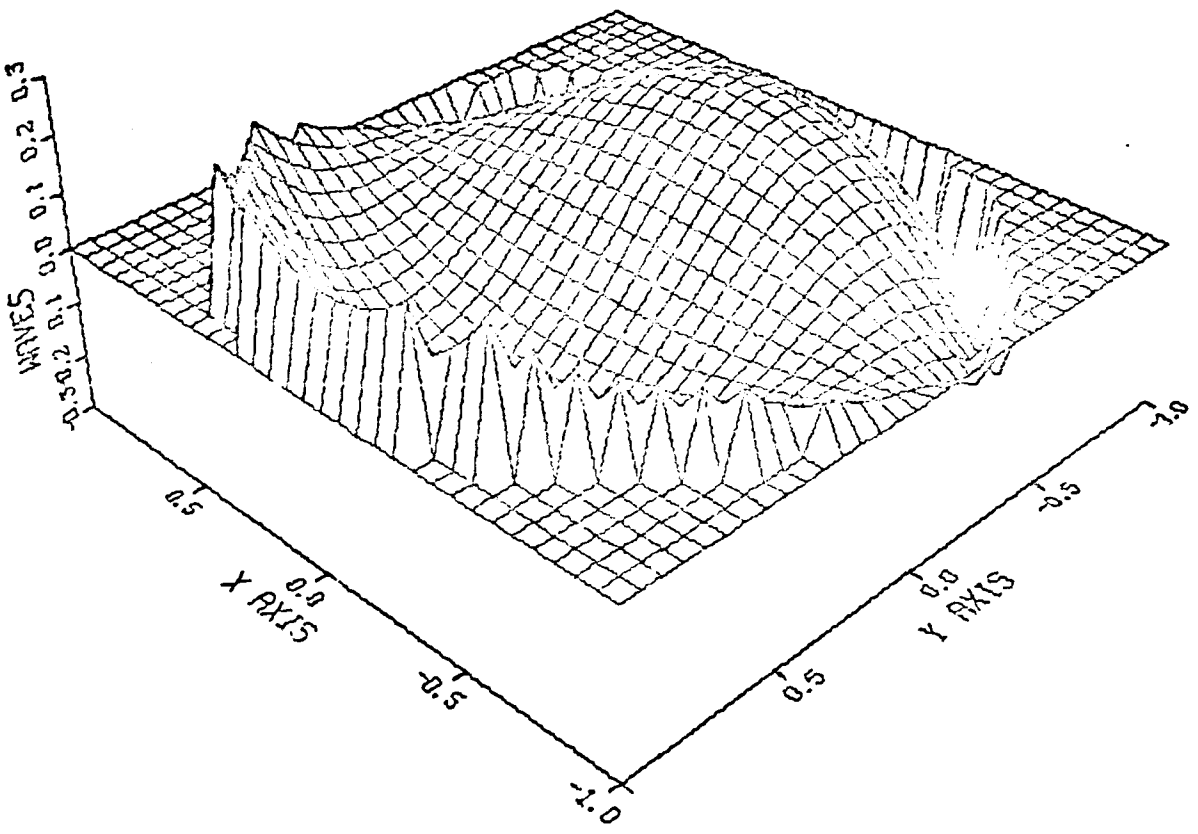
ZERNIKE COEF 5, SET AT 0.1



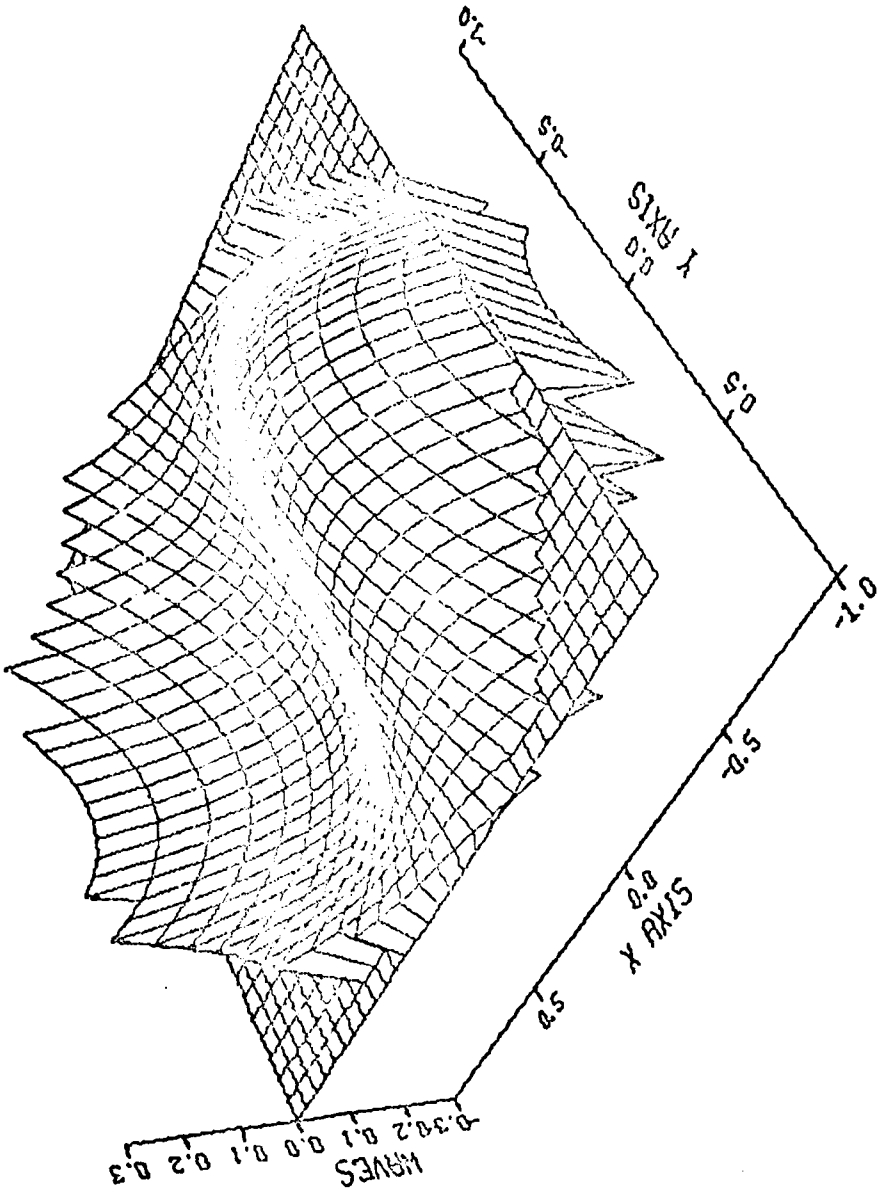
ZERNIKE COEF 6, SET AT 0.1



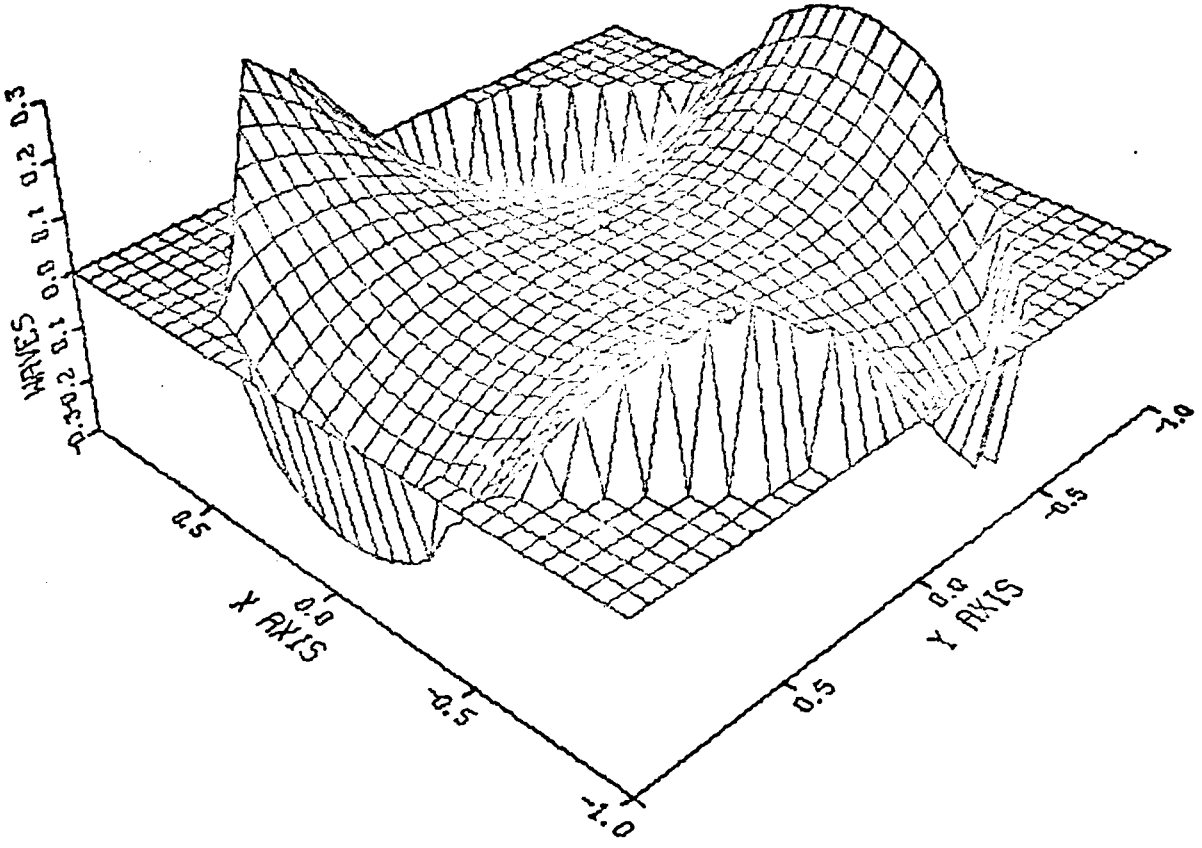
ZERNIKE COEF 7, SET AT 0.1



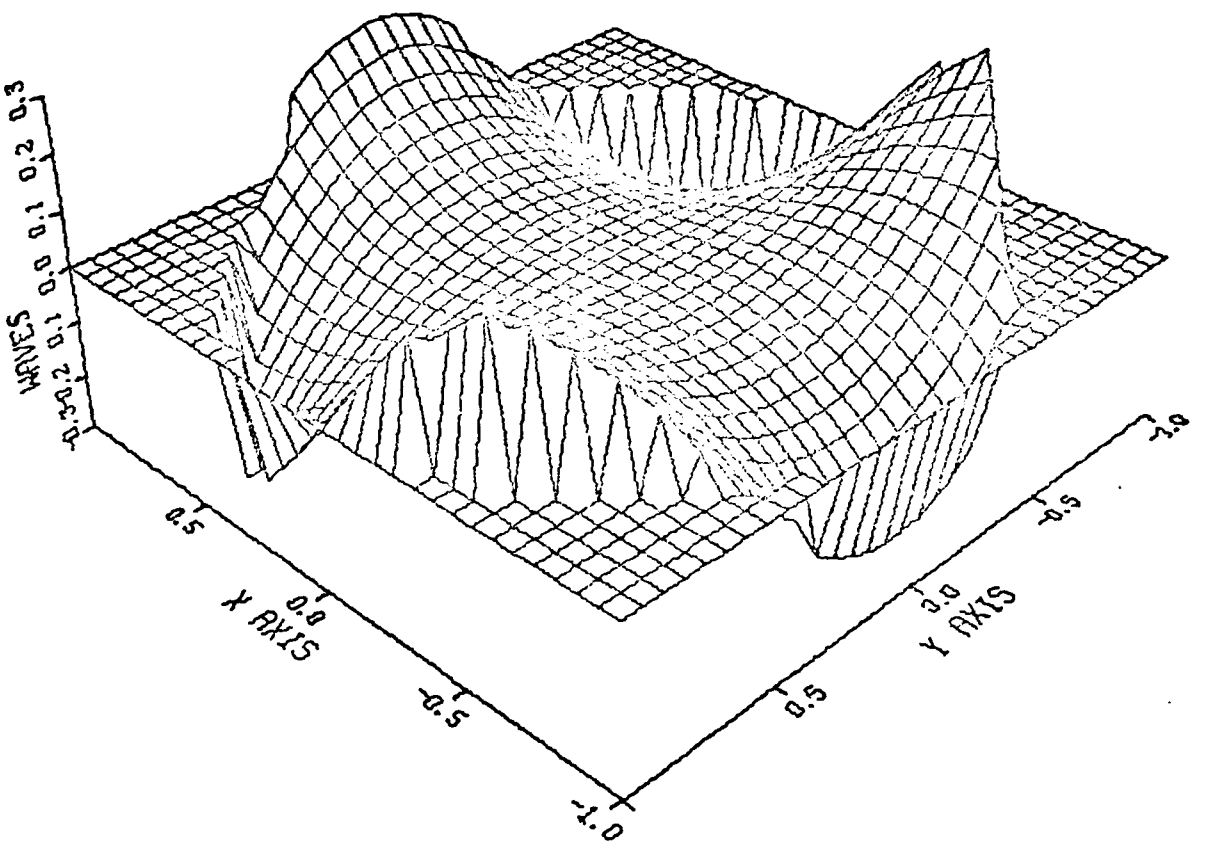
ZERNIKE COEF 8, SET AT 0.1



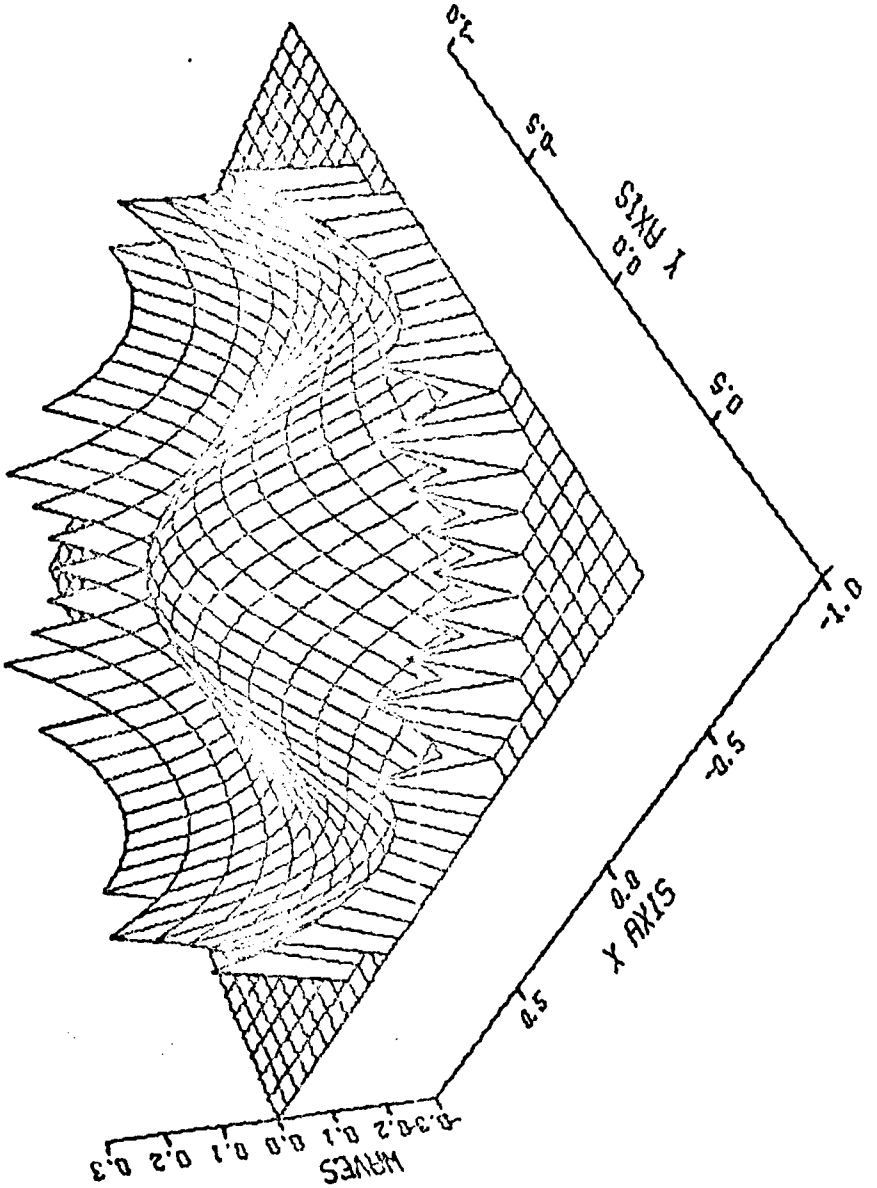
ZERNIKE COEF 9, SET AT 0.1



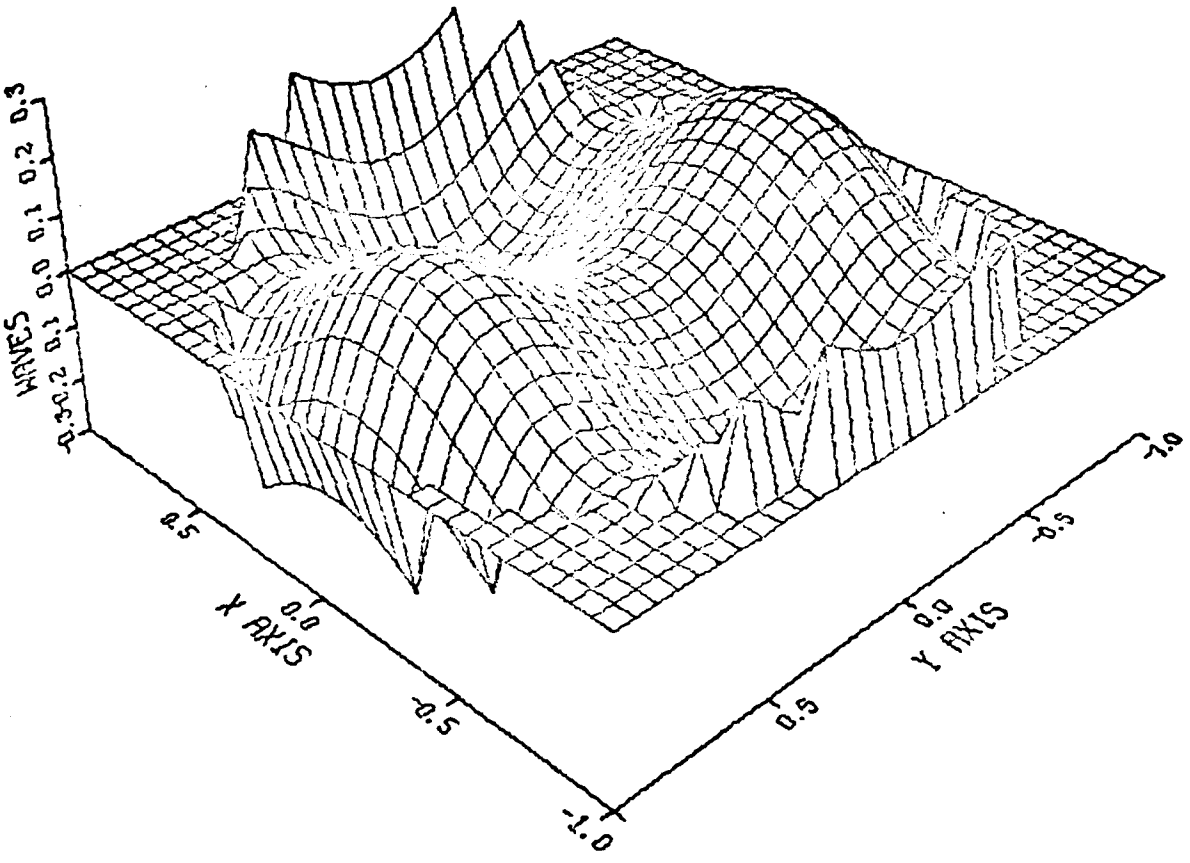
ZERNIKE COEF 10, SET AT 0.1



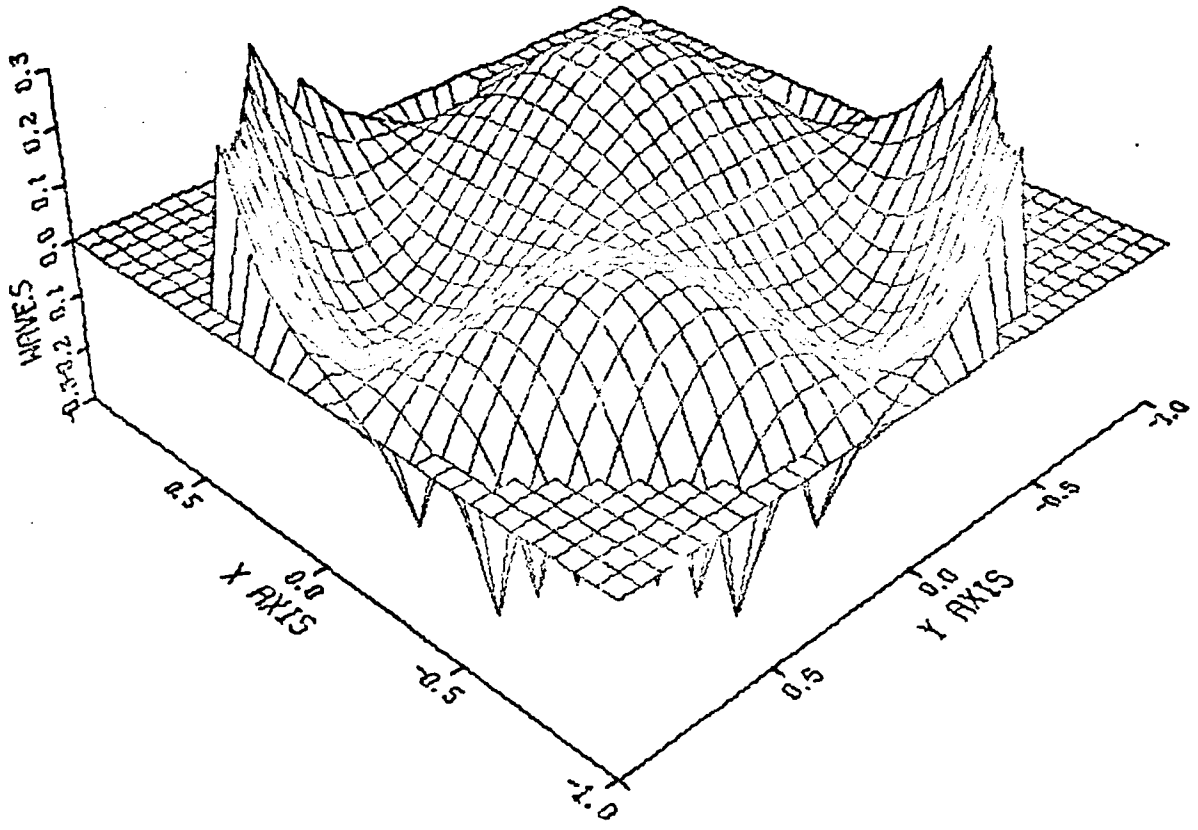
ZERNIKE COEF 11, SET AT 0.1



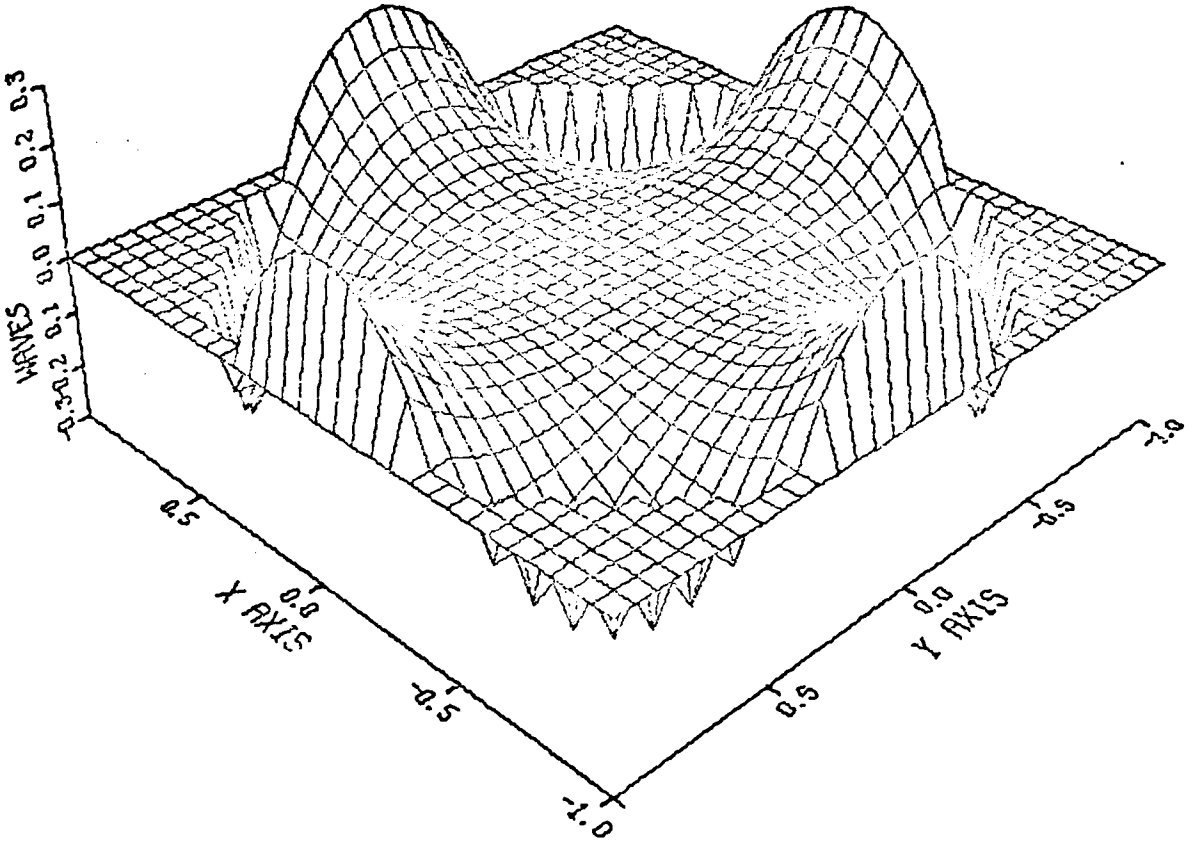
ZERNIKE COEF 12, SET AT 0.1



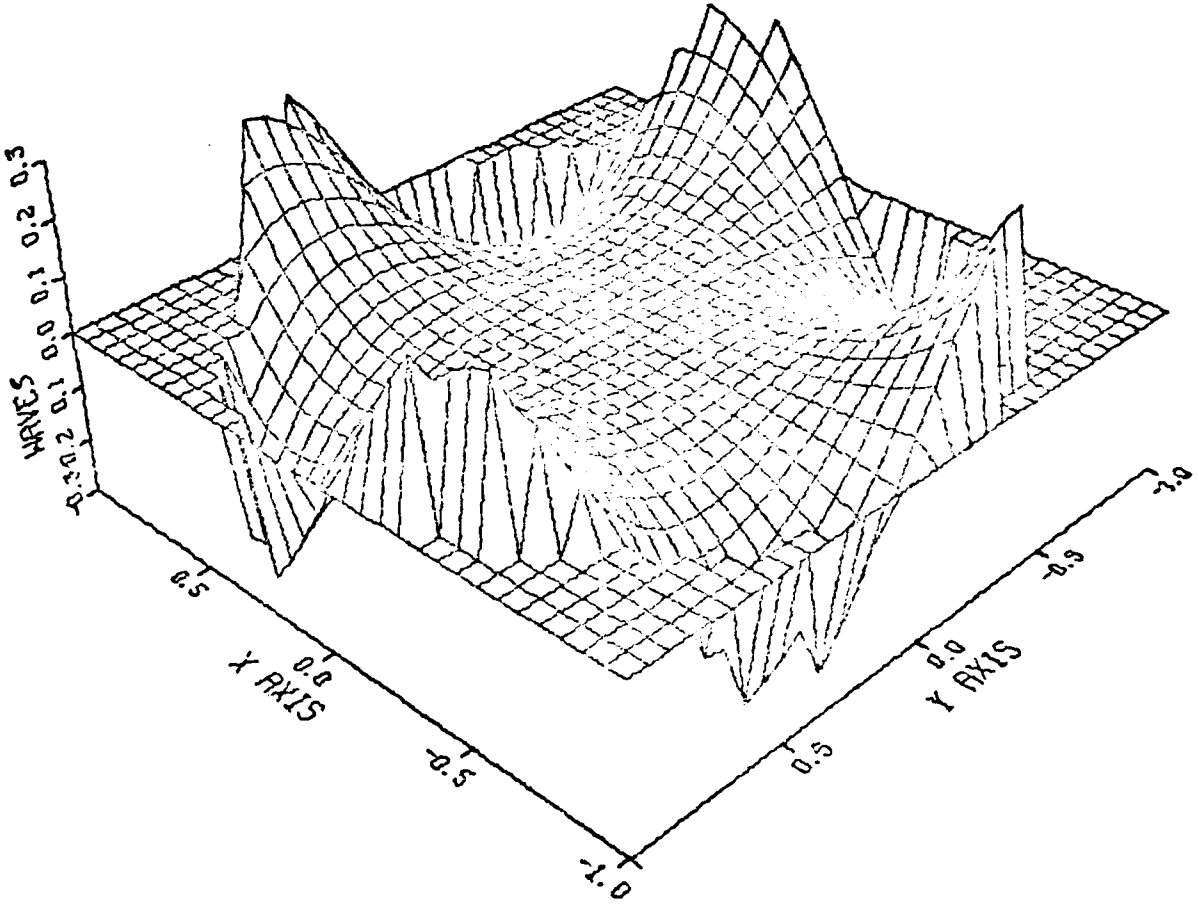
ZERNIKE COEF 13, SET AT 0.1



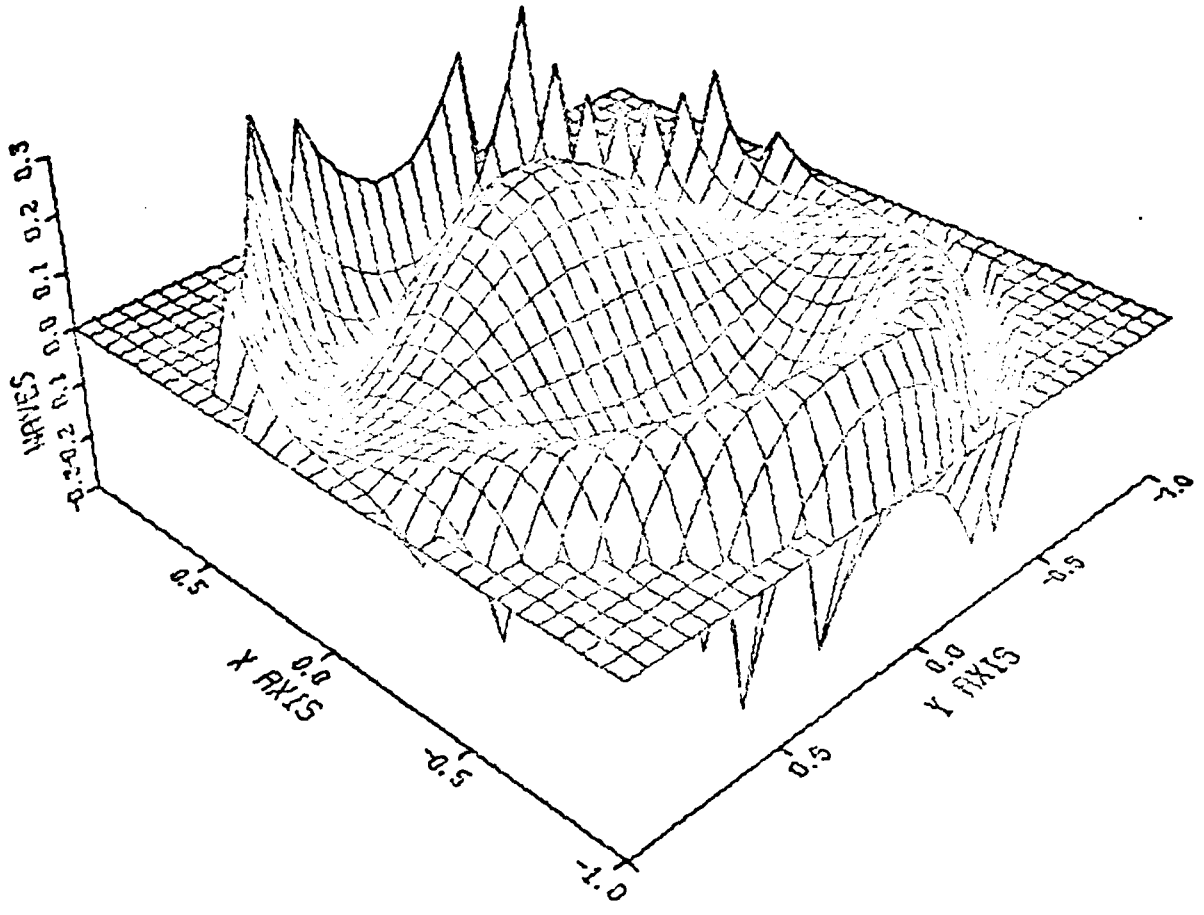
ZERNIKE COEF 14, SET AT 0.1



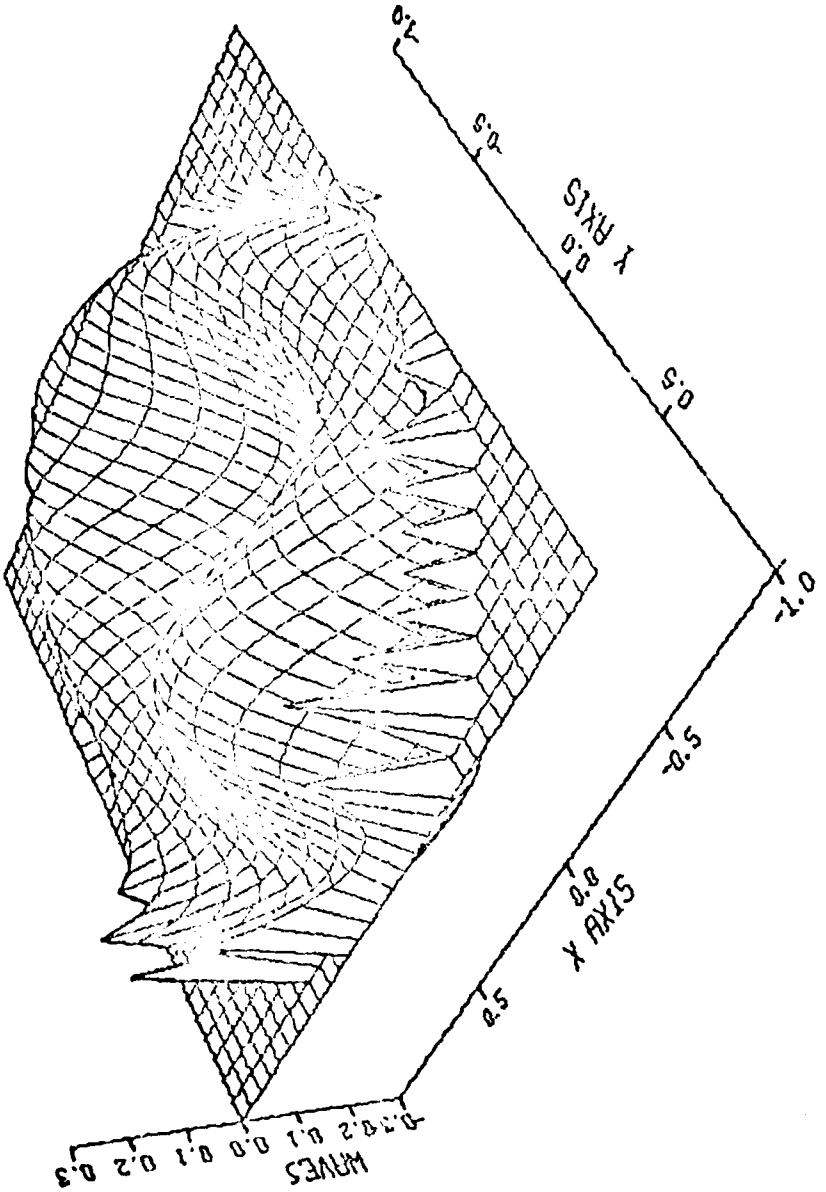
ZERNIKE COEF 15, SET AT 0.1



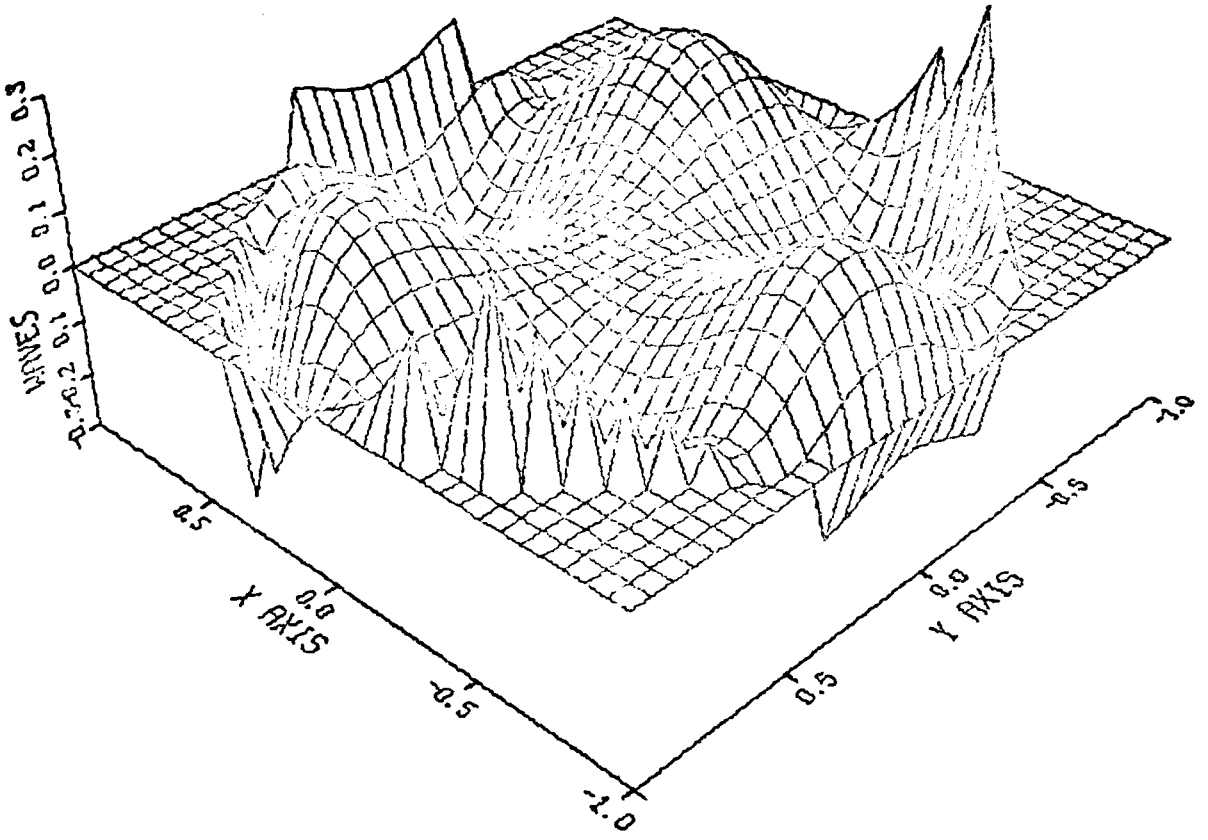
ZERNIKE COEF 16, SET AT 0.1



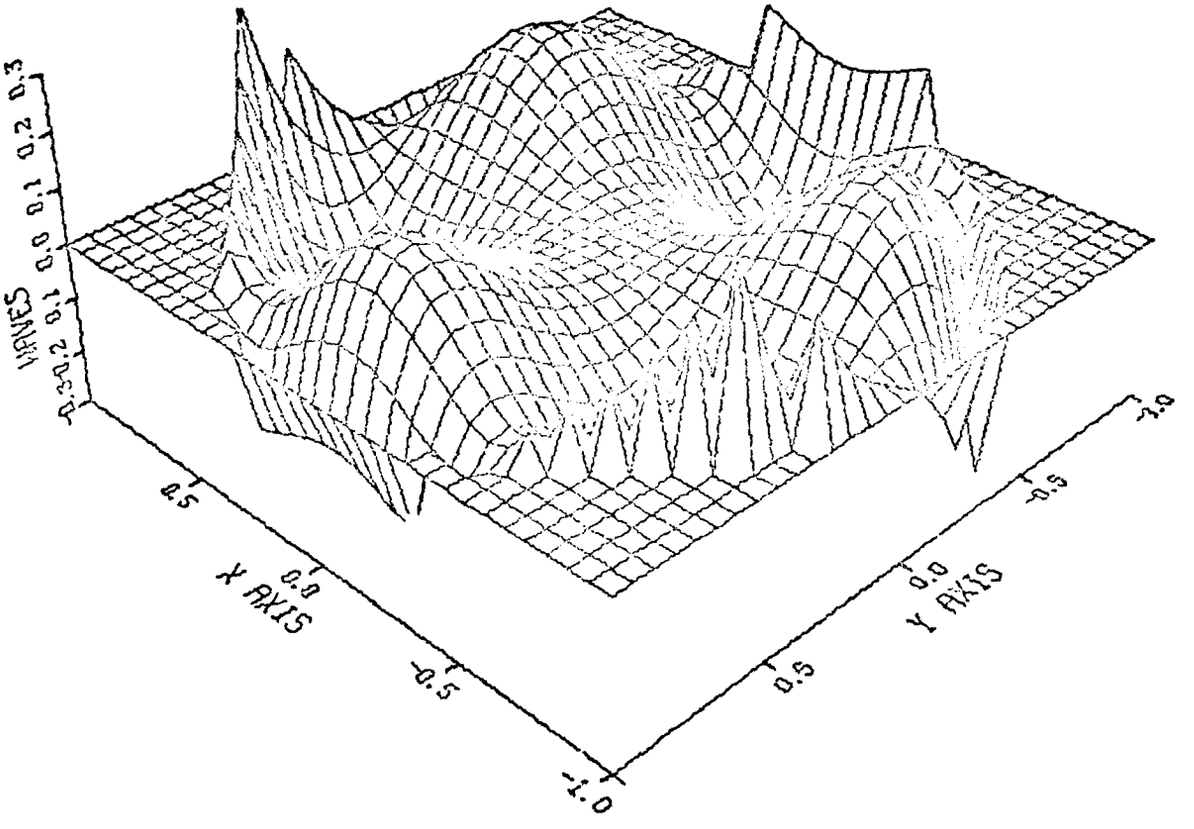
ZERNIKE COEF 17, SET AT 0.1



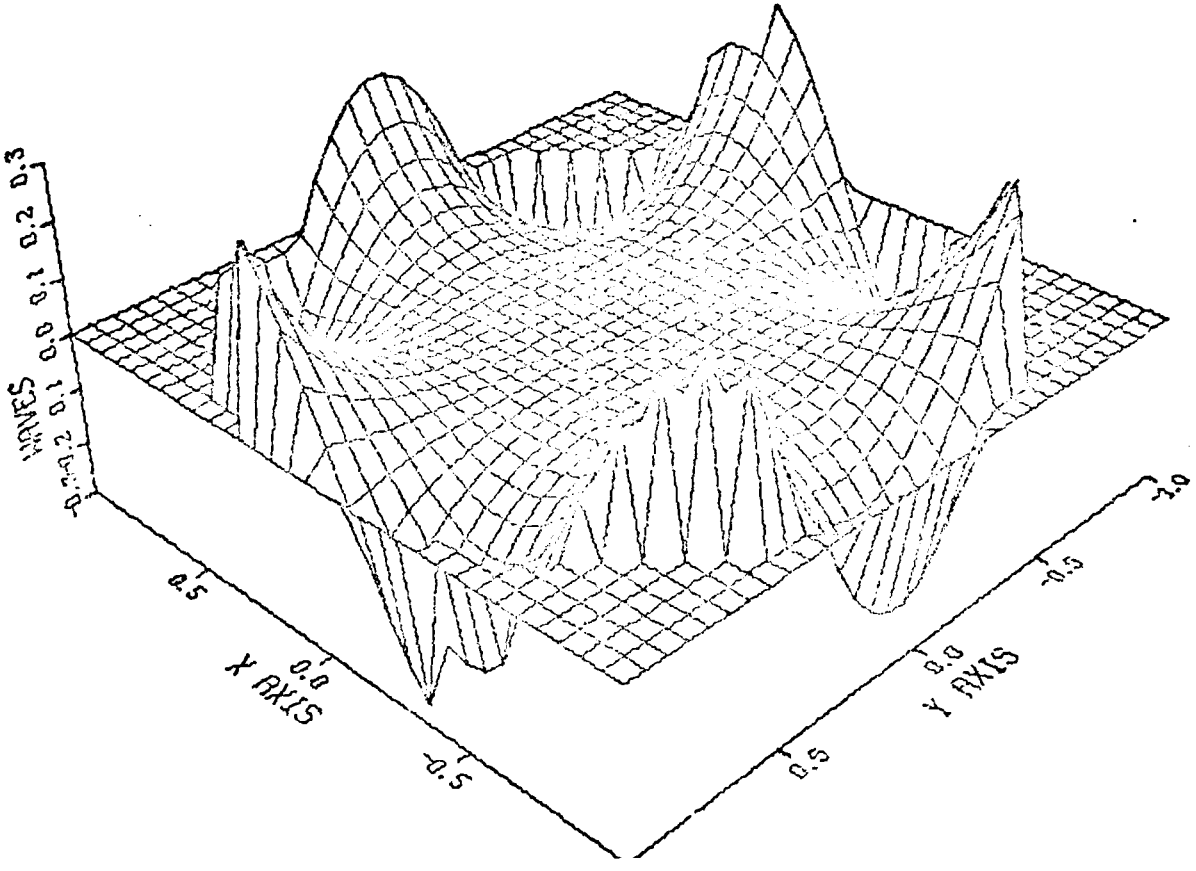
ZERNIKE COEF 18, SET AT 0.1



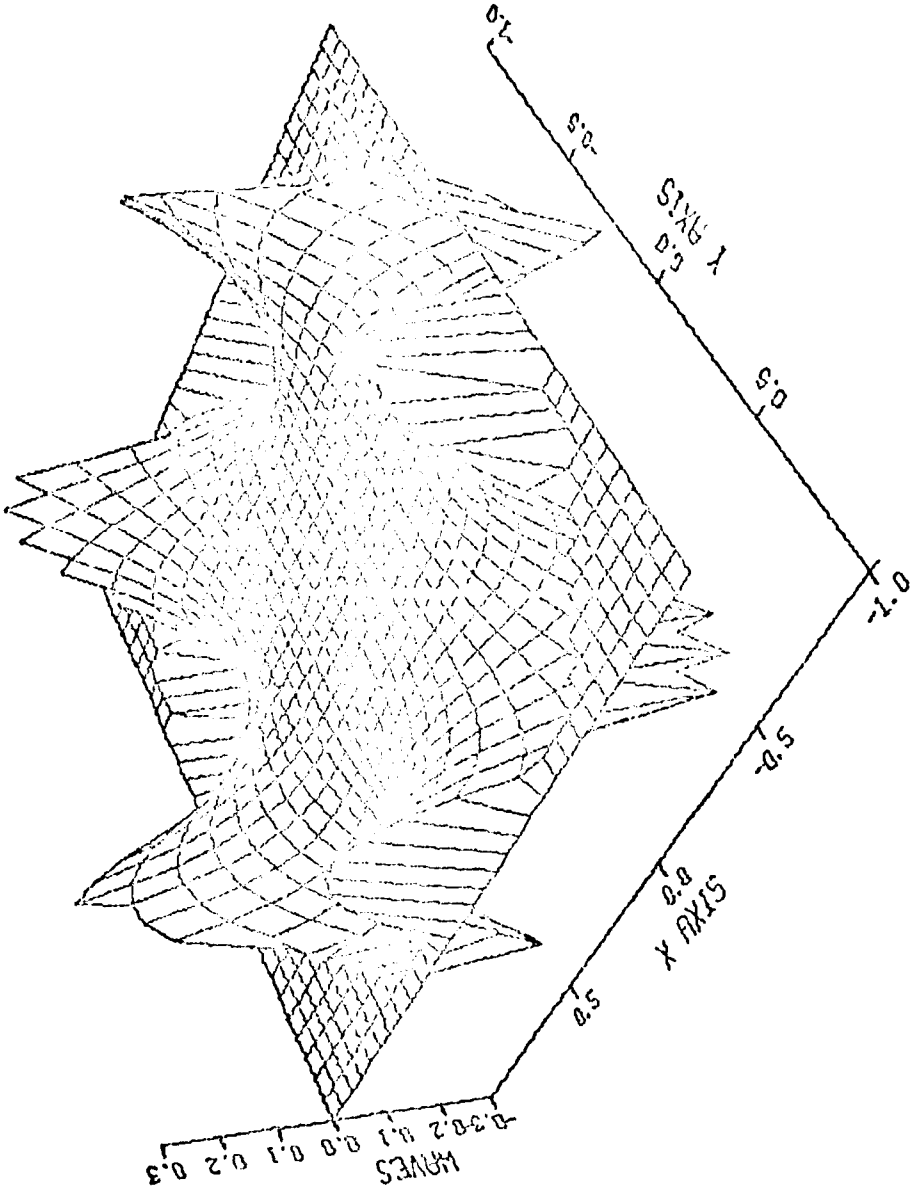
ZERNIKE COEF 19, SET AT 0.1



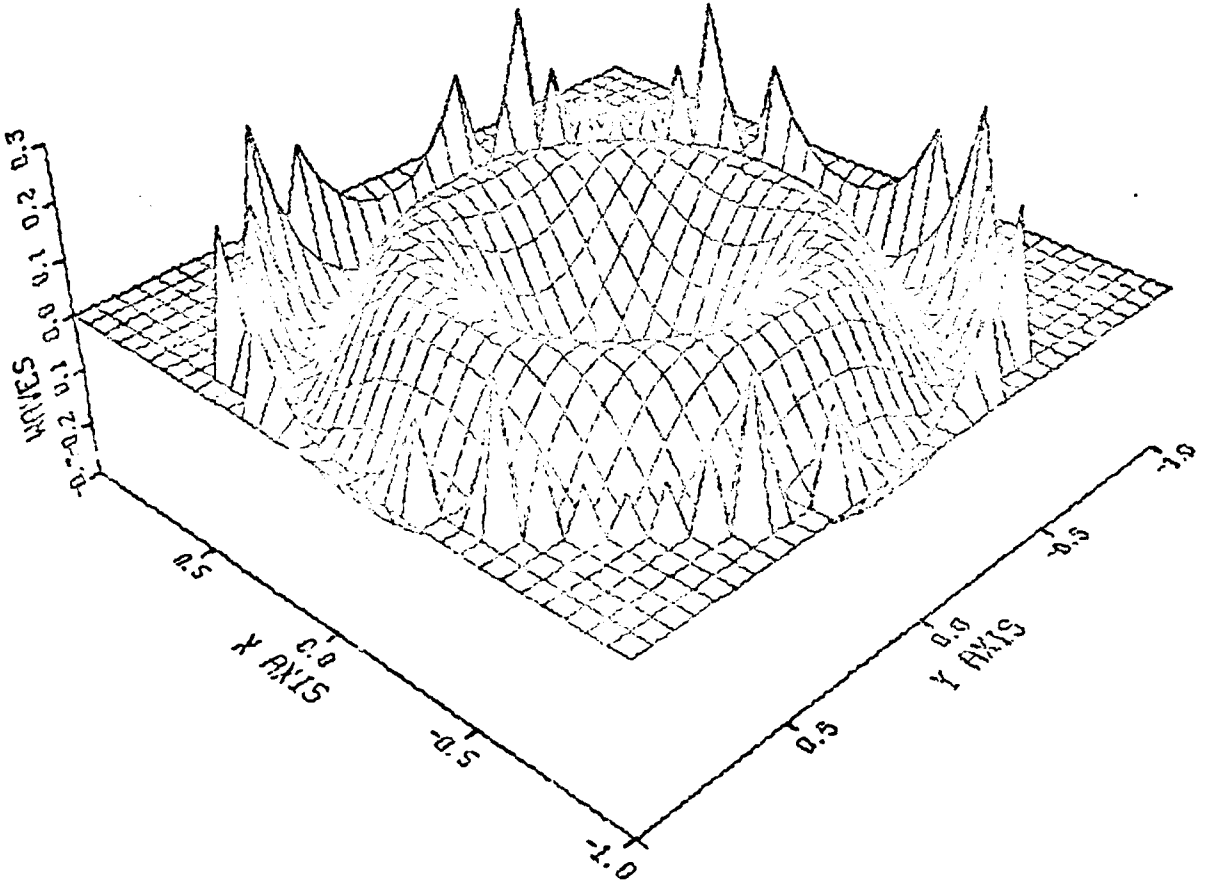
ZERNIKE COEF 20, SET AT 0.1



ZERNIKE COEF 21, SET AT 0.1



ZERNIKE COEF 22, SET AT 0.1



Appendix C

Zernike Polynomial Development

Appendix C

Zernike Polynomial Development

The basic theory behind the development of the Zernike polynomials comes from the works presented by Born and Wolf (Ref.1;Appendix VII). The basic idea presented by Born and Wolf was to find a set of polynomials which are invariant with respect to rotation. In general one wishes the following condition to be met

$$\iint_{x^2+y^2\leq 1} V_a^*(x,y)V_b(x,y)dxdy = A_{ab}\delta_{ab} \qquad (C.1)$$

where V is the general polynomial, the "*" denotes complex conjugate, δ is the Kronecker delta, and A is a normalization constant (Ref.1:767). The rotational invariance is expressed as

$$V(x,y) = G(\theta)V(x',y') \qquad (C.2)$$

where

$$\begin{aligned} x' &= x\cos\theta + y\sin\theta \\ y' &= x\sin\theta + y\sin\theta \end{aligned} \qquad (C.3)$$

and $G(\theta)$ is continuous with a period of 2π and $G(0)\equiv 1$. G must also satisfy the condition of

$$G(\theta_1)G(\theta_2) = G(\theta_1+\theta_2) \tag{C.4}$$

since the rotation through one angle, then the next angle is the same as the rotation through the sum of both angles. A simple solution for a function to represent $G(\theta)$ is

$$G(\theta) = e^{il\theta} \tag{C.5}$$

where l is an integer. Putting Equation (C.5) into Equation (C.2) and setting $x'=r$ and $y'=0$, and using Equation (C.3) yields

$$V(r\cos\theta, r\sin\theta) = R(r)e^{il\theta} \tag{C.6}$$

where $R(r)$ is strictly a function of r. Born and Wolf then show how if V is a polynomial of degree n then R must have the same degree n but with a power of r not less than l. Thus, "the set of the Zernike circle polynomials is distinguished from all other such sets by the property that it contains a polynomial for each pair of permissible values of n (degree) and l (angular dependence)." (Ref.1:768)(Their emphasis) Thus V can be expressed as

$$V_n^l(r\cos\theta, r\sin\theta) = R_n^l(r)e^{il\theta} \tag{C.7}$$

From Equation (C.1,7), $R_n^l(r)$ must satisfy the relation

$$\int_0^1 R_n^l(r)R_{n'}^l(r)r\,dr = a_n^l\delta_{nn'} \tag{C.8}$$

where

$$a_n^1 = \frac{A_n^1}{2\pi}$$

With this result, the basic form of the Zernike polynomial is expressed.

Born and Wolf then find an explicit expression for $R_n^m(r)$ as

$$R_n^m(r) = t^{m/2} Q_{\frac{n-m}{2}}(t) \qquad (C.9)$$

where $t=r$ and $Q_{\frac{n-m}{2}}(t)$ is a polynomial in t of degree $\frac{1}{2}(n-m)$. Since Q must still satisfy the condition of R, Q must satisfy Equation (C.8) or

$$\tfrac{1}{2}\int_0^1 t^m Q_k(t) Q_{k'}(t)dt = a_n^n \delta_{kk'} \qquad (C.10)$$

where $k=\frac{1}{2}(n-m)$ and $k'=\frac{1}{2}(n'-m)$. Born and Wolf then equate Q with the Jacobi polynomials. Through substitution and normalization procedures, the final form of the Radial polynomial is

$$R_n^m(r) = \sum_{s=0}^{\frac{1}{2}(n-m)} (-1)^s \frac{(n-s)!\, r^{n-2s}}{s!(\frac{1}{2}(n+m)-s)!(\frac{1}{2}(n-m)-s)!} \qquad (C.11)$$

Thus, using the explicit expression for R in Equation (C.11) and expanding the angular function G into sine and cosine terms, the set of Zernike polynomials can be found.

Appendix D

Flowcharts of Software

Appendix D

Flowcharts of Software

Contained in this appendix are flowcharts of all of the routines in the software used in the analysis of annular wave-fronts. The flowcharts appear in the following order: Main program, GAMSUB, FAPER, VALID, EDGE, ZRAD, ZANG, CONTUR, RMSERR, SERPRNT, INVERT, MULT, ARPR1, and ARPR2. The explanation of these routines appears in Chapter IV, and the listing of the program is in Appendix E. Figure 36 defines all of the symbols used in the flowcharts. The first appearance of a variable will be defined in the upper left hand corner of the page it first appears on.

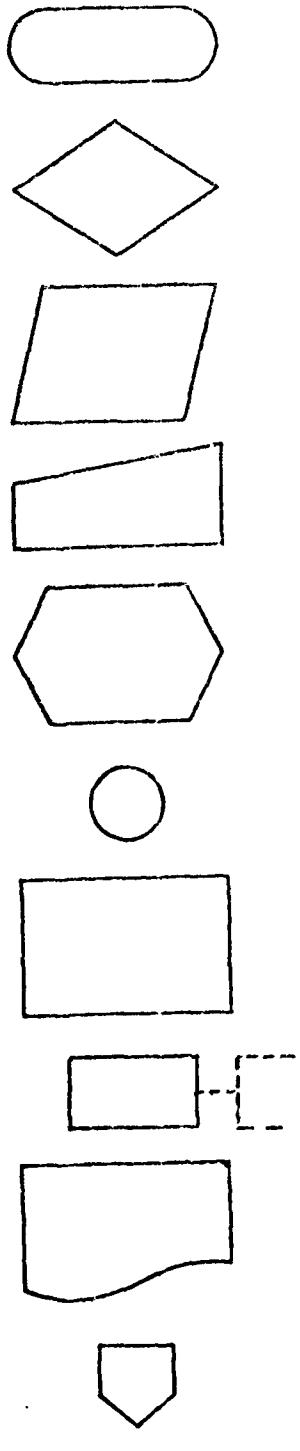
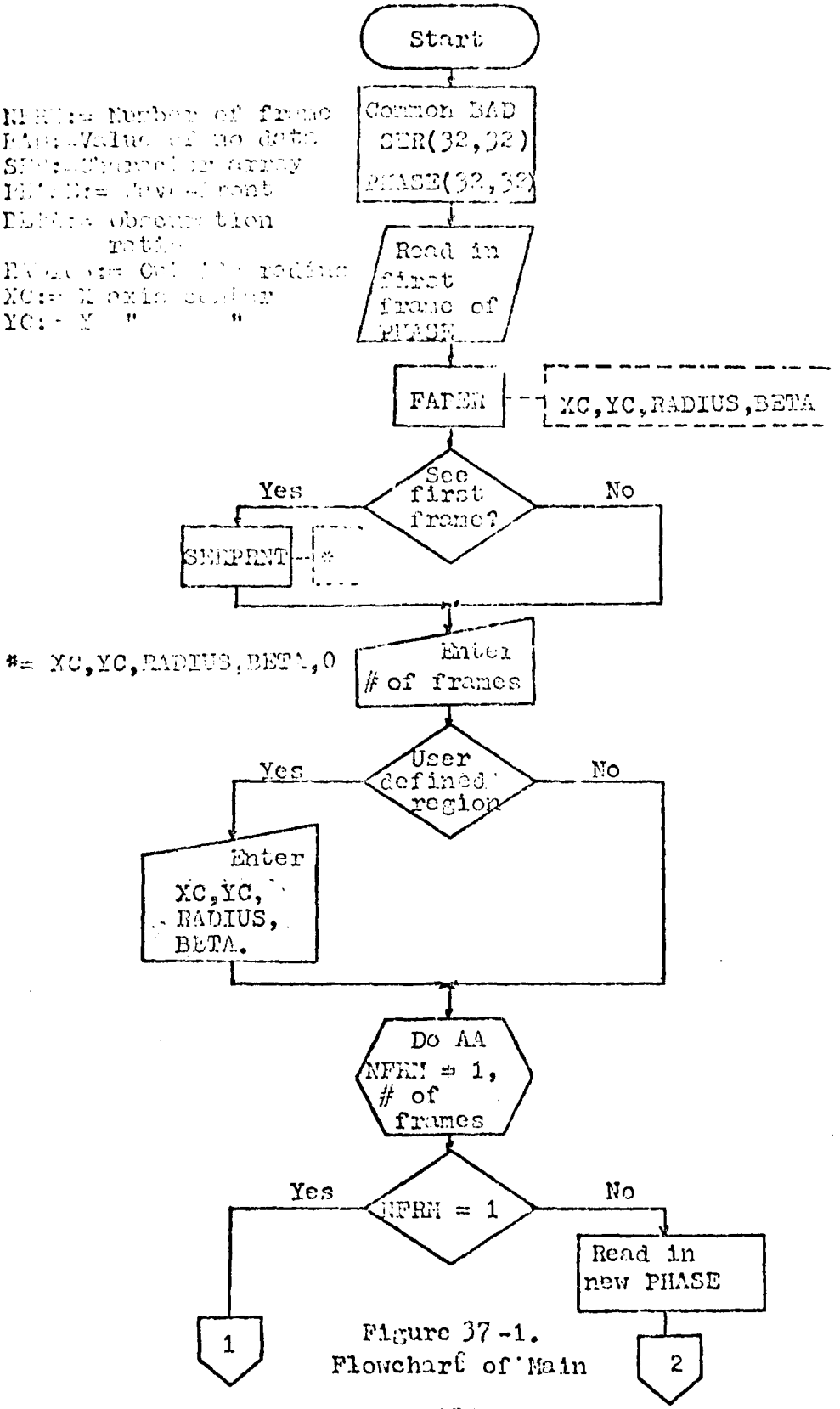


Figure 36. Flowchart Symbol Definitions



*= XC,YC,RADIUS,BETA,0

Figure 37-1.
Flowchart of Main

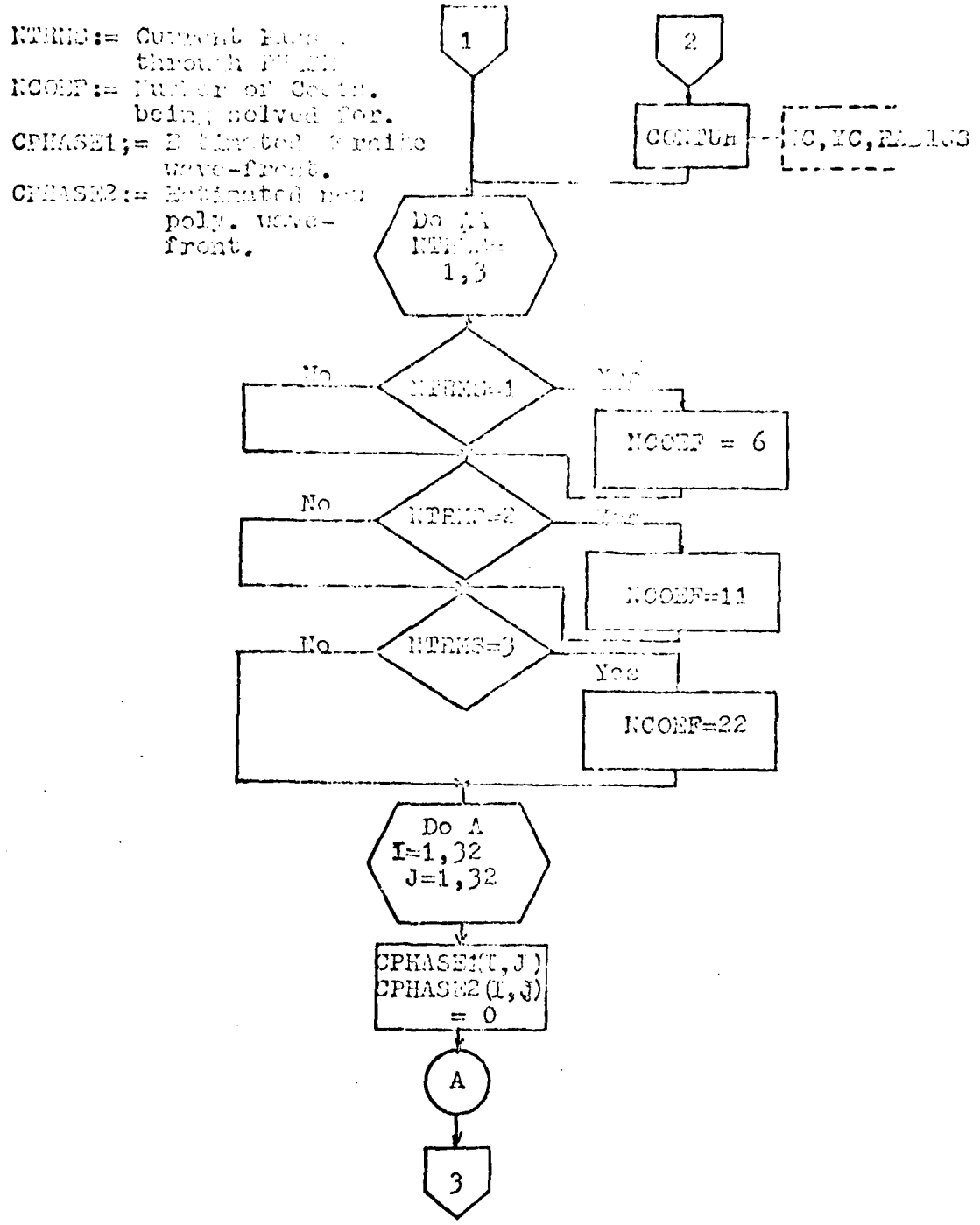


Figure 37-2. Flowchart of Main

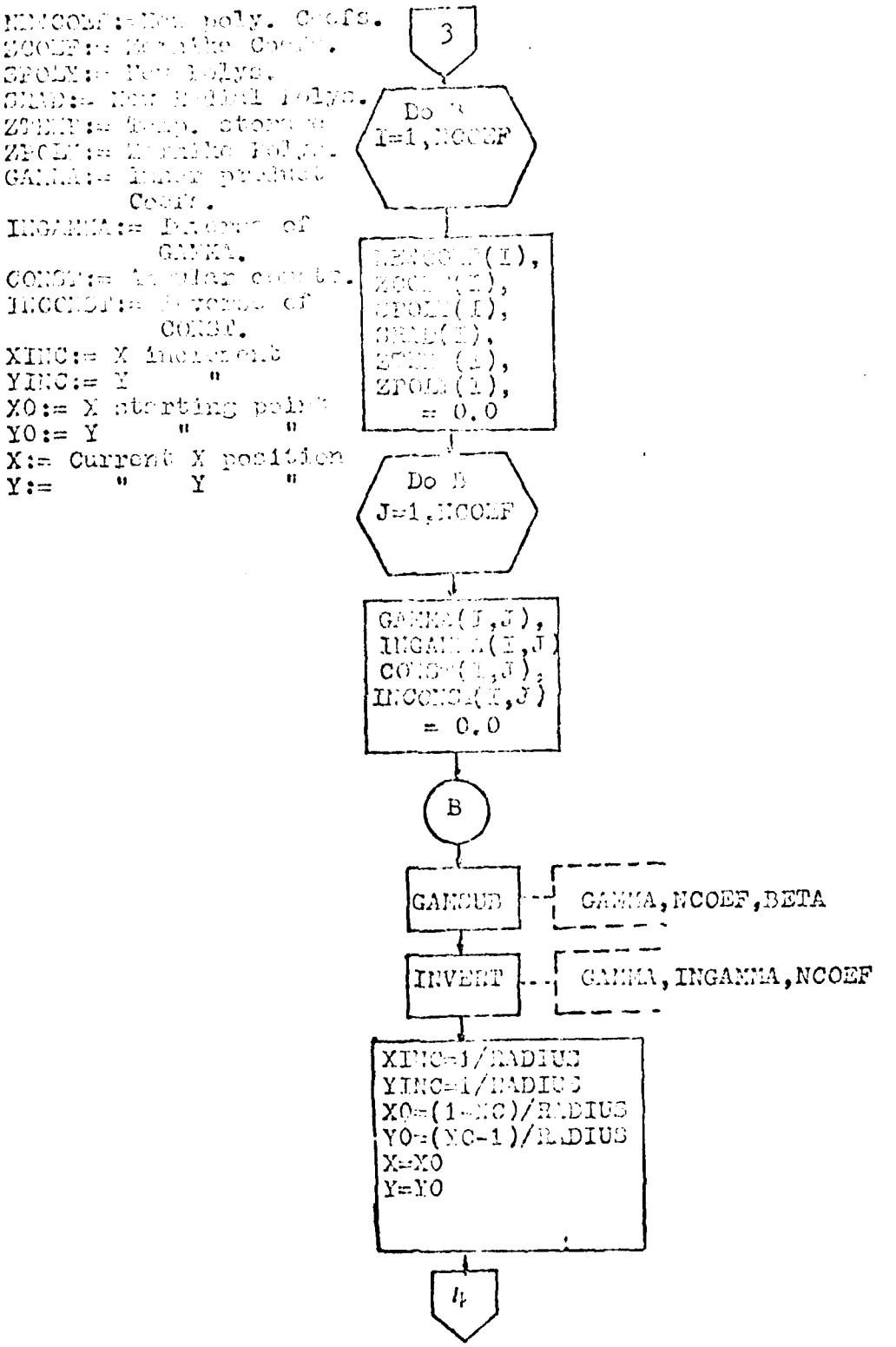


Figure 37-3. Flowchart of Main

TOTAL:= Total number of valid points in search area.
ZRADIAL:= Zernike Radial Poly.

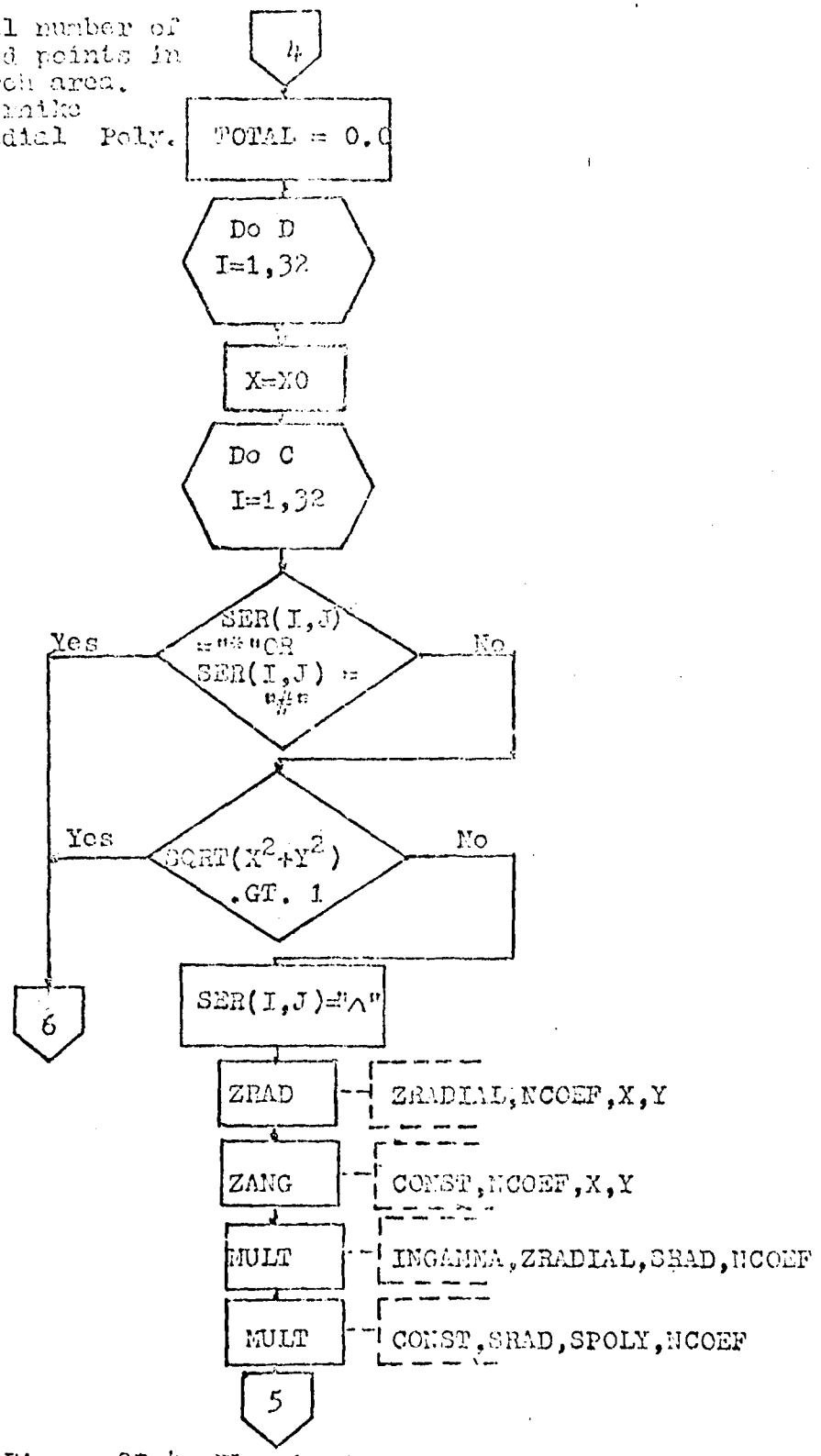


Figure 37-4. Flowchart of Main

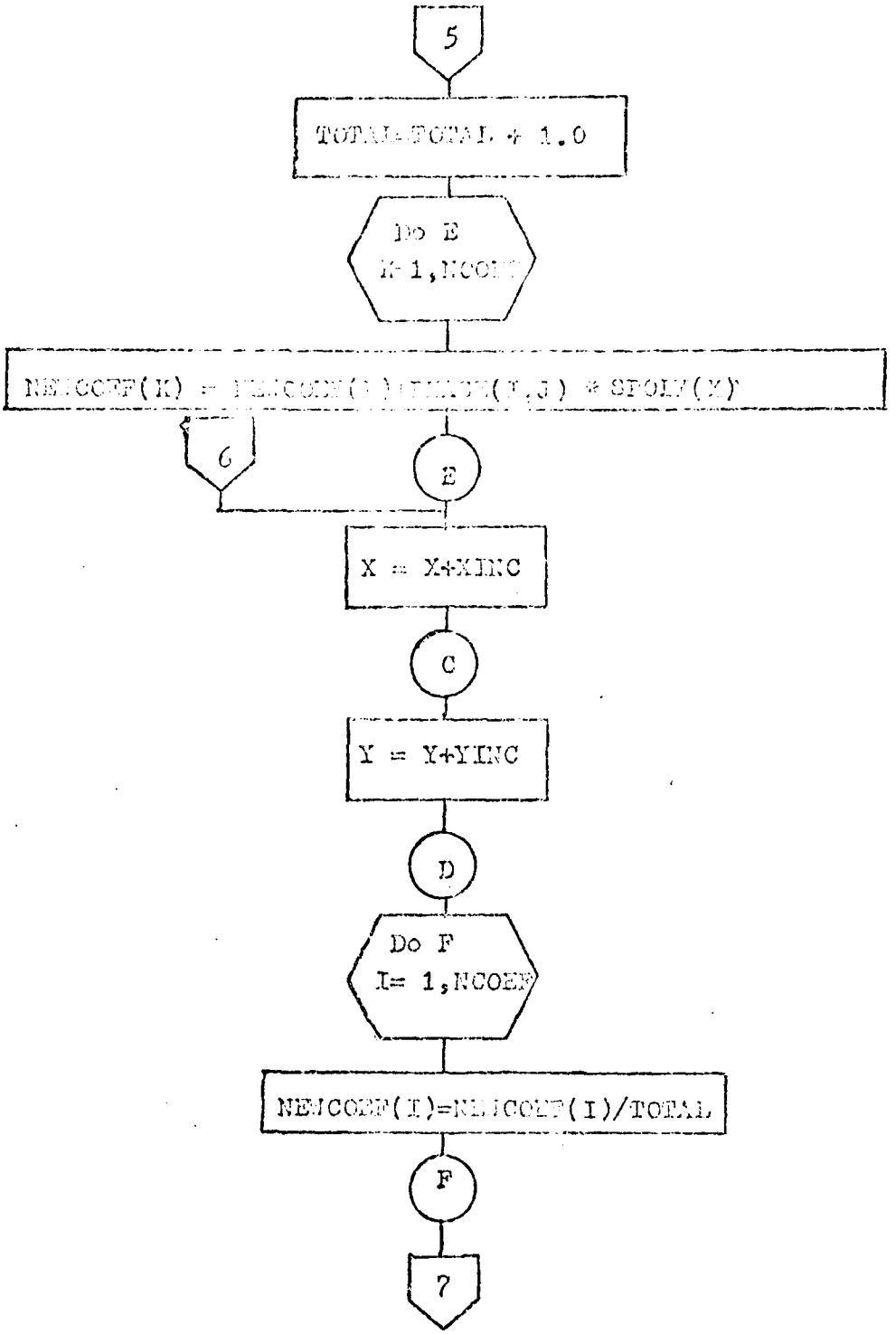


Figure 37-5. Flowchart of Main

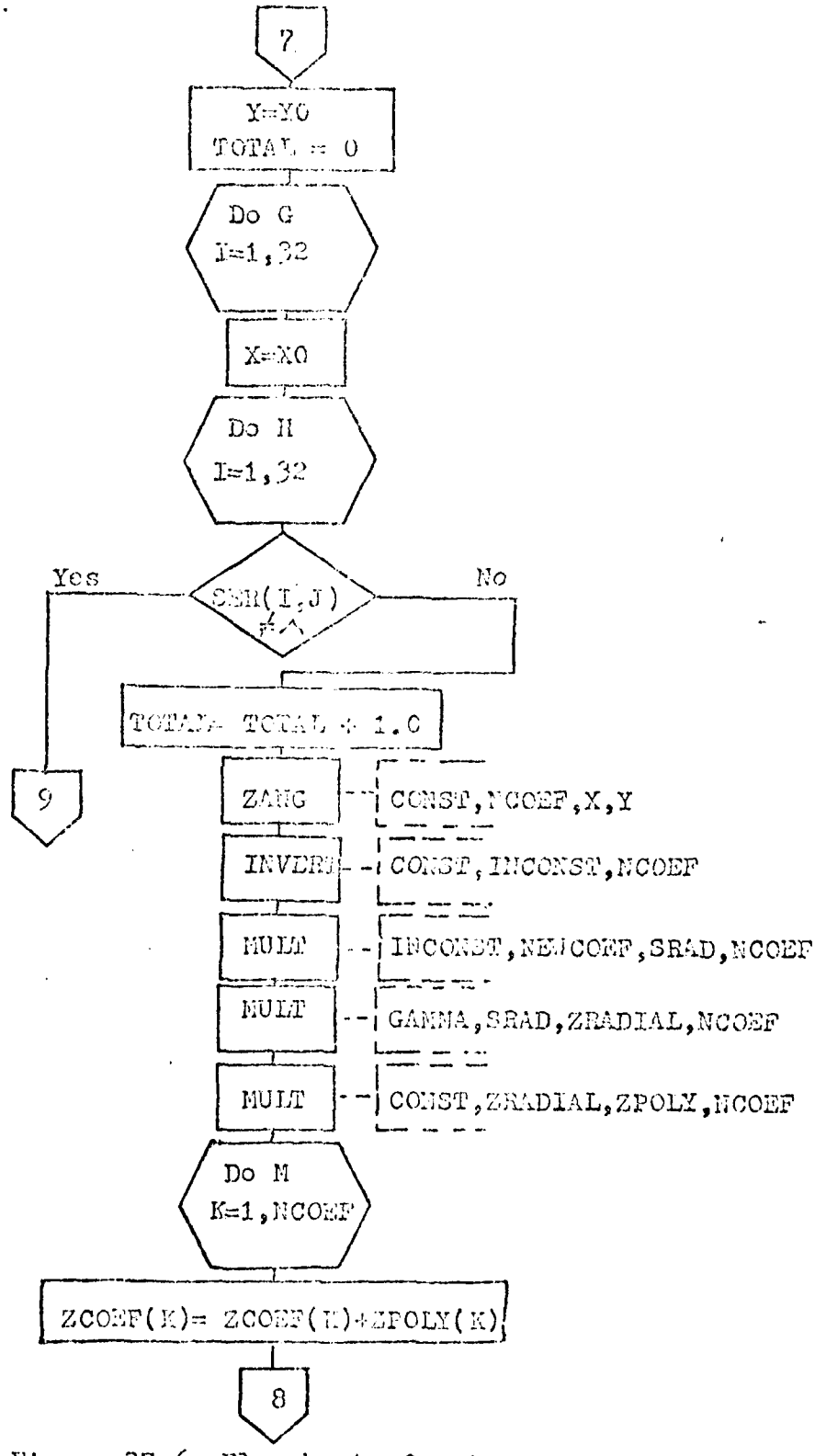


Figure 37-6. Flowchart of Main

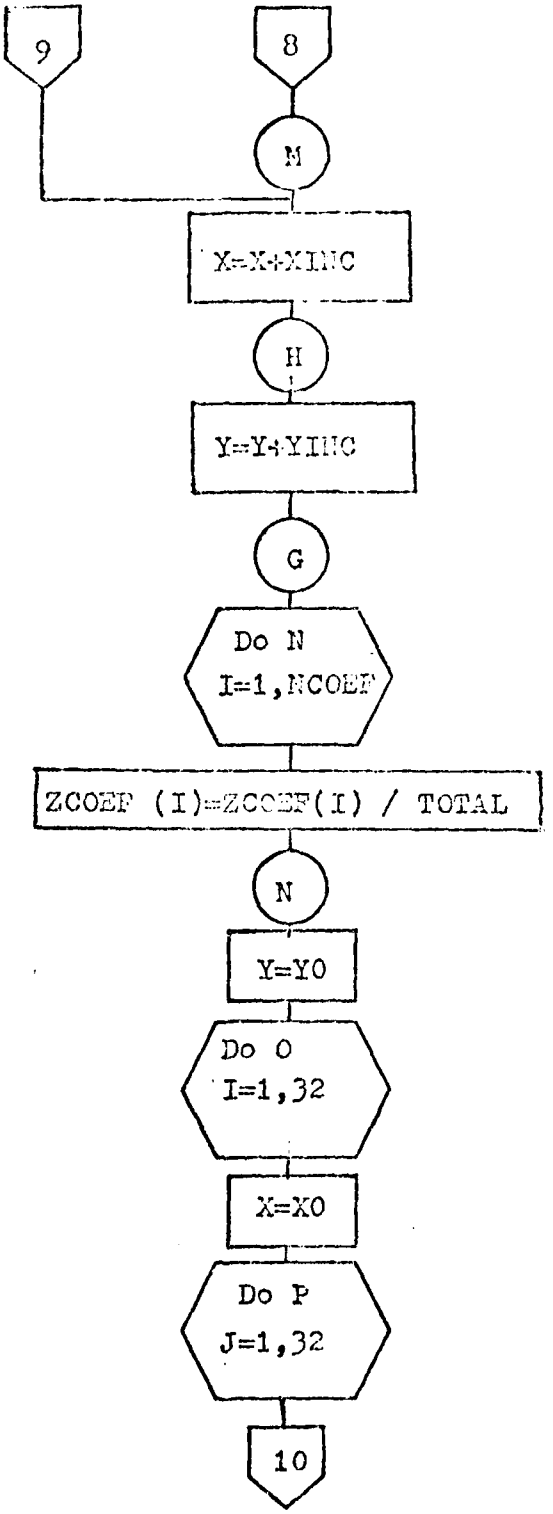


Figure 37-7. Flowchart on Main

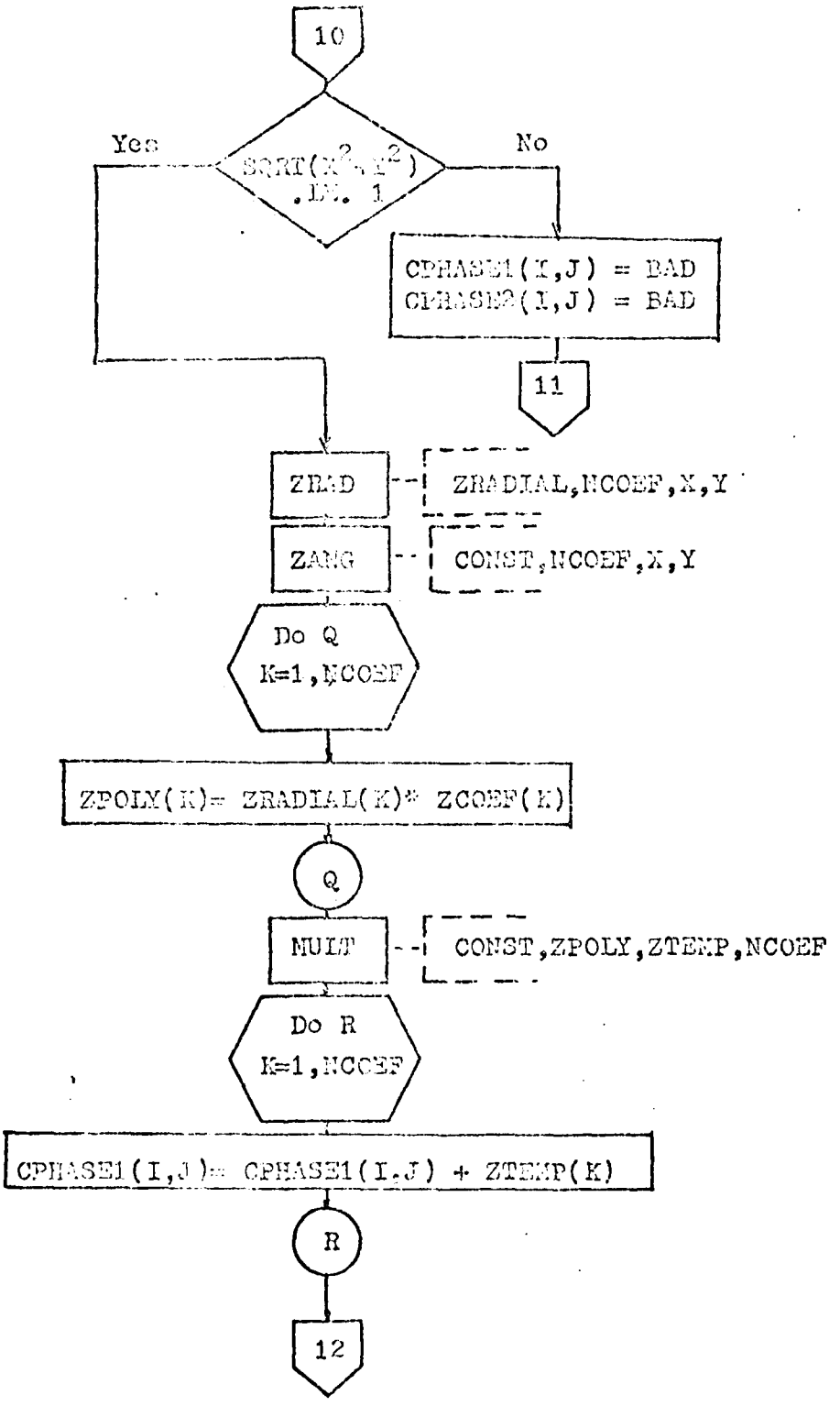


Figure 37-8. Flowchart of Main

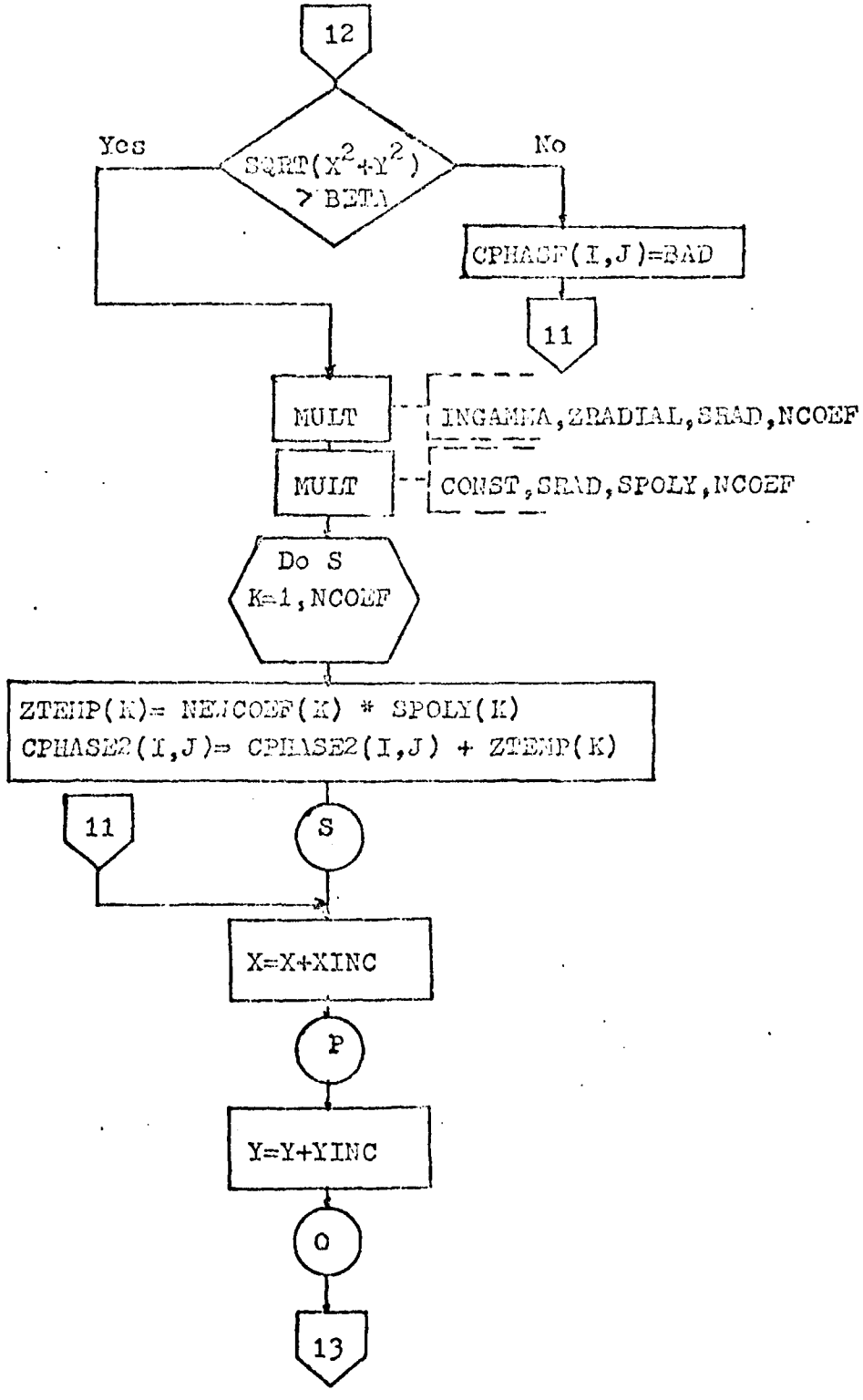


Figure 37-9. Flowchart of Main

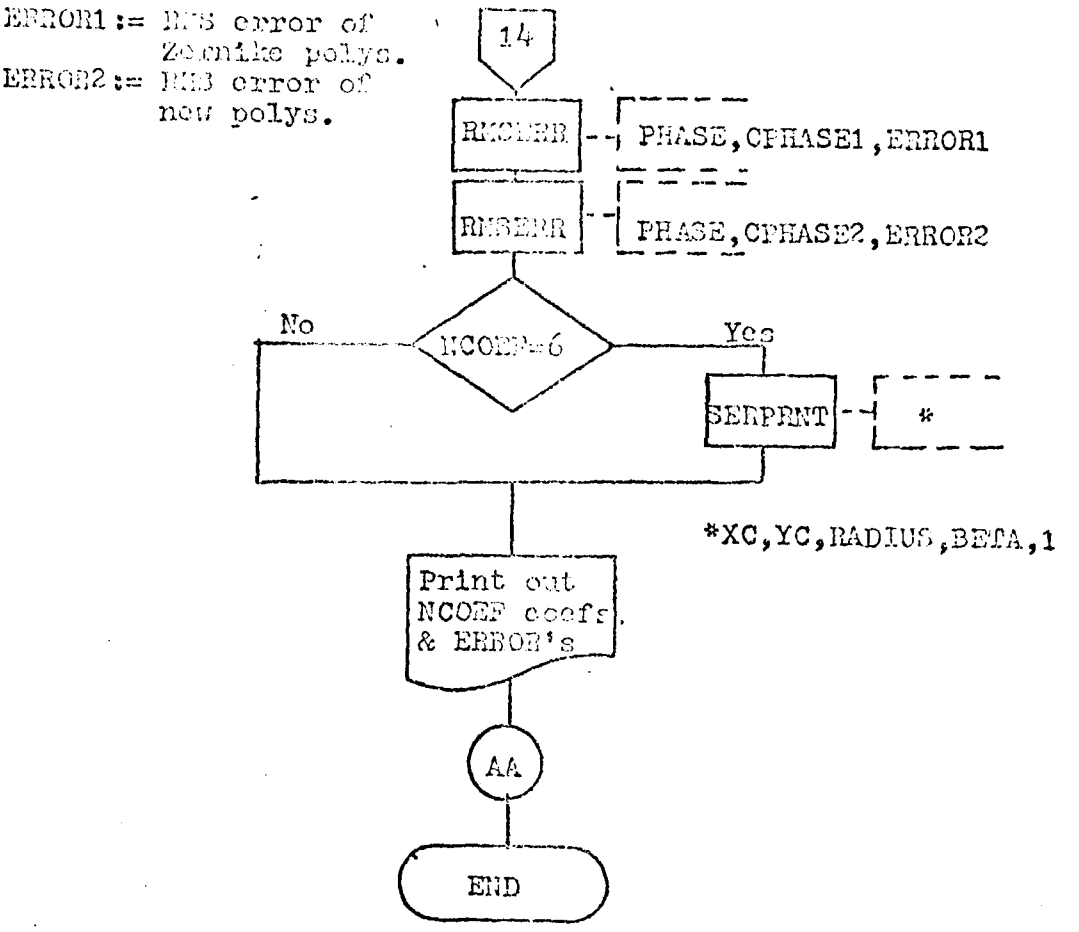


Figure 37-10. Flowchart of Main

Gmnj:= γ_{nj}^{m}

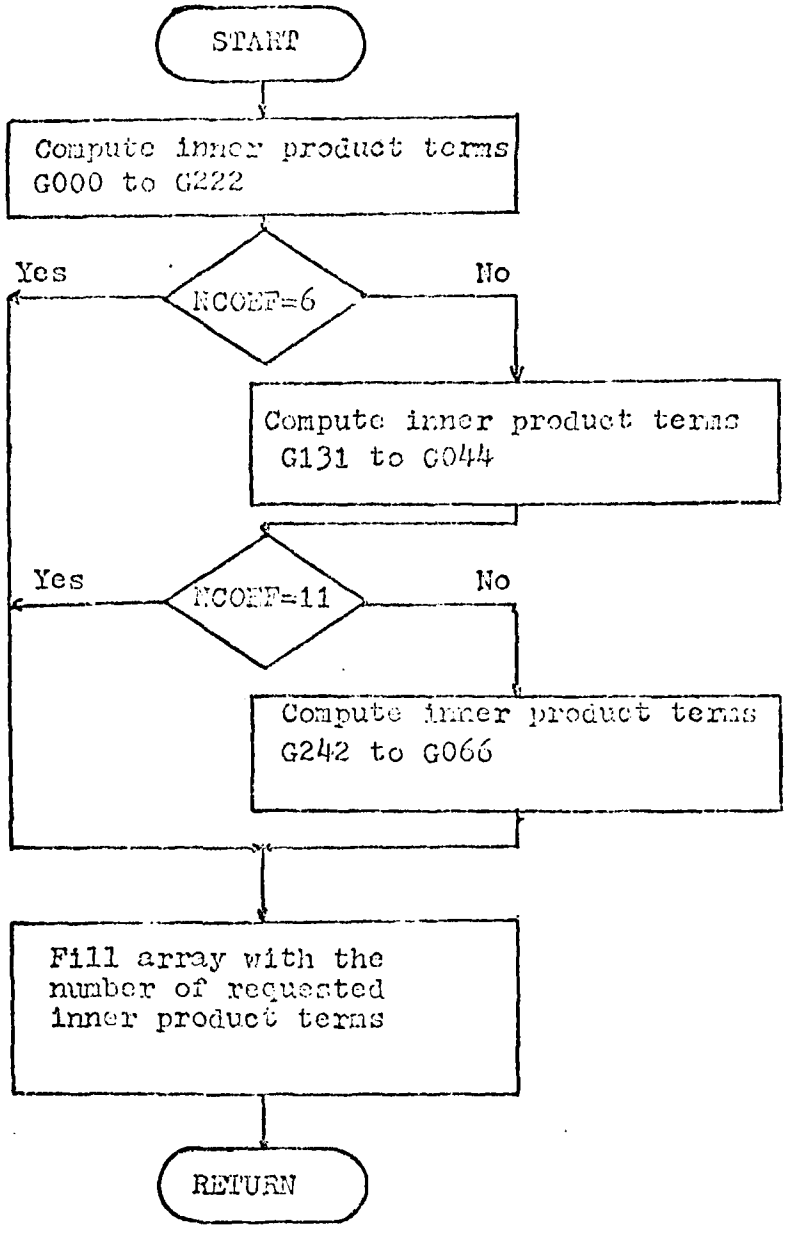


Figure 38. Flowchart of GAMSUB

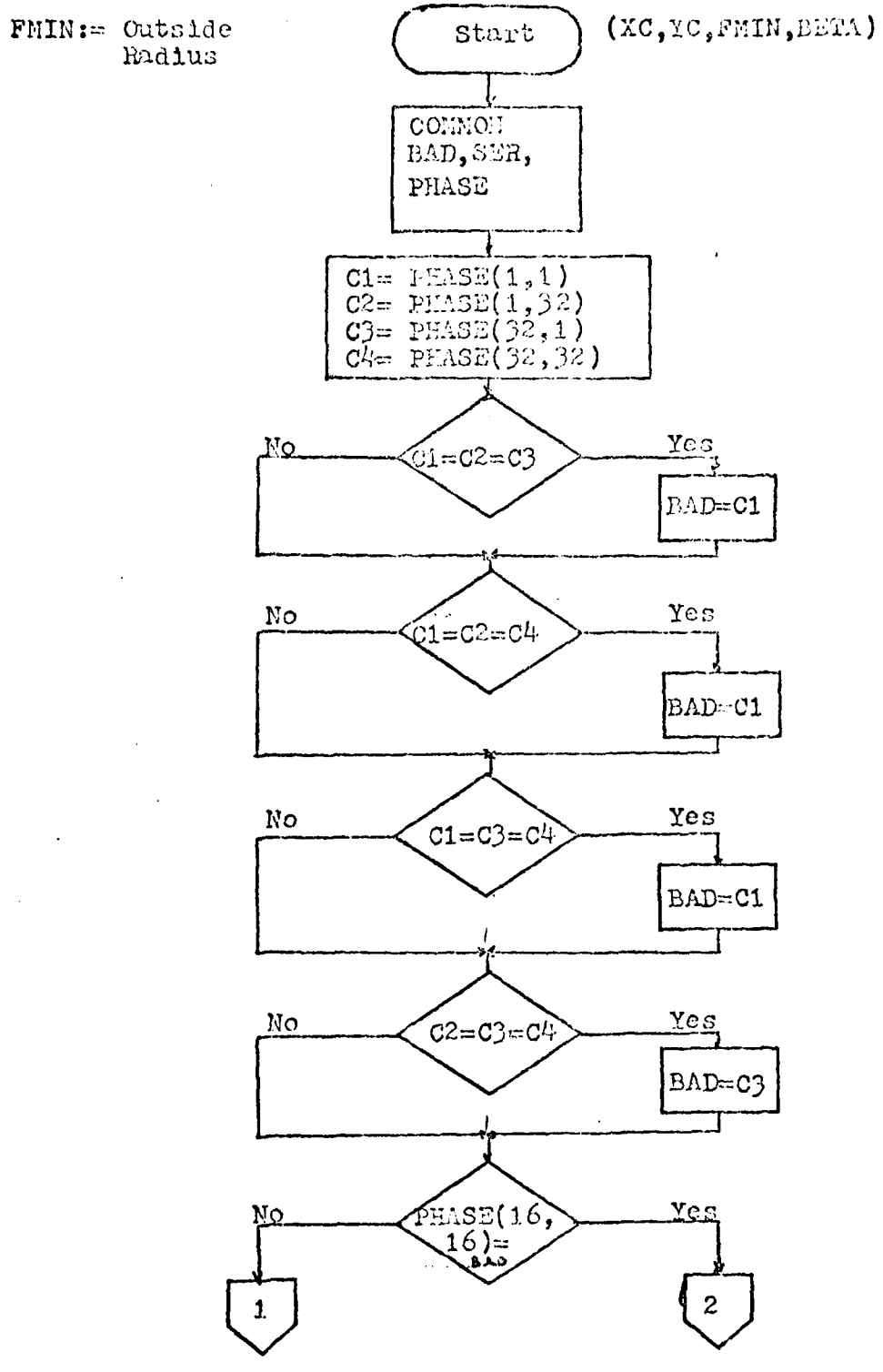


Figure 39-1. Flowchart of FAPER

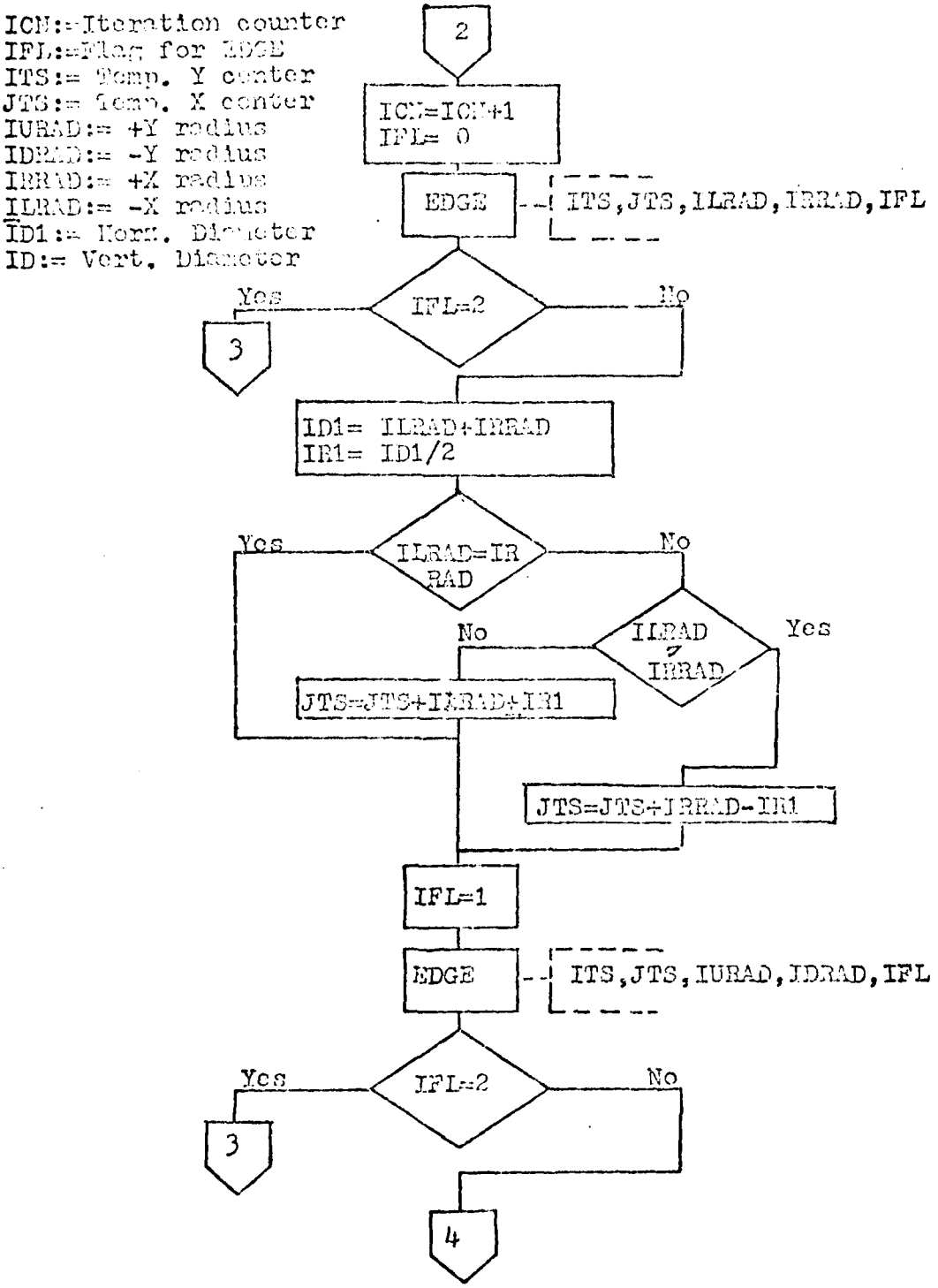


Figure 39-2. Flowchart of FAPER

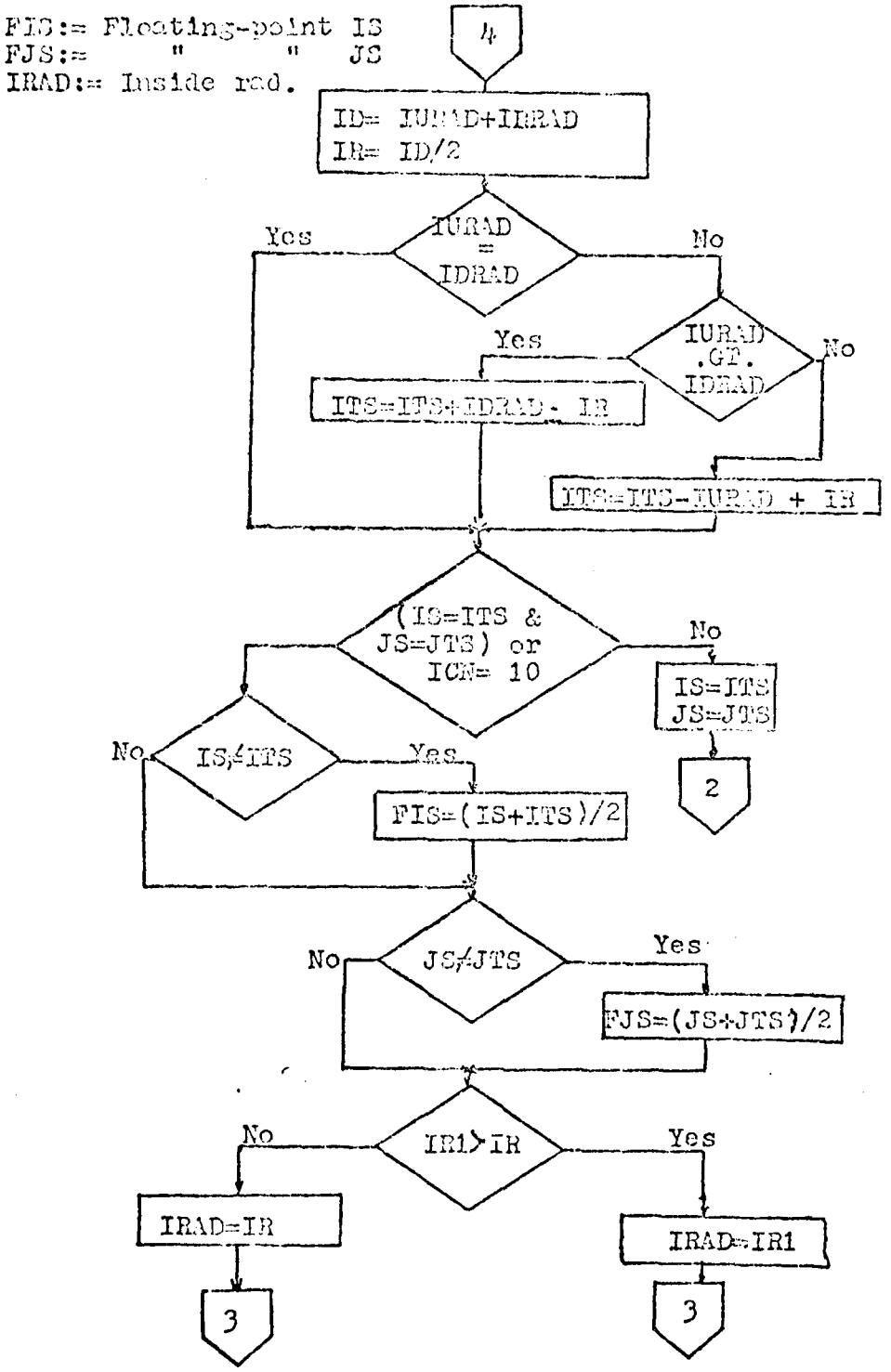


Figure 39-3. Flowchart of FAPER

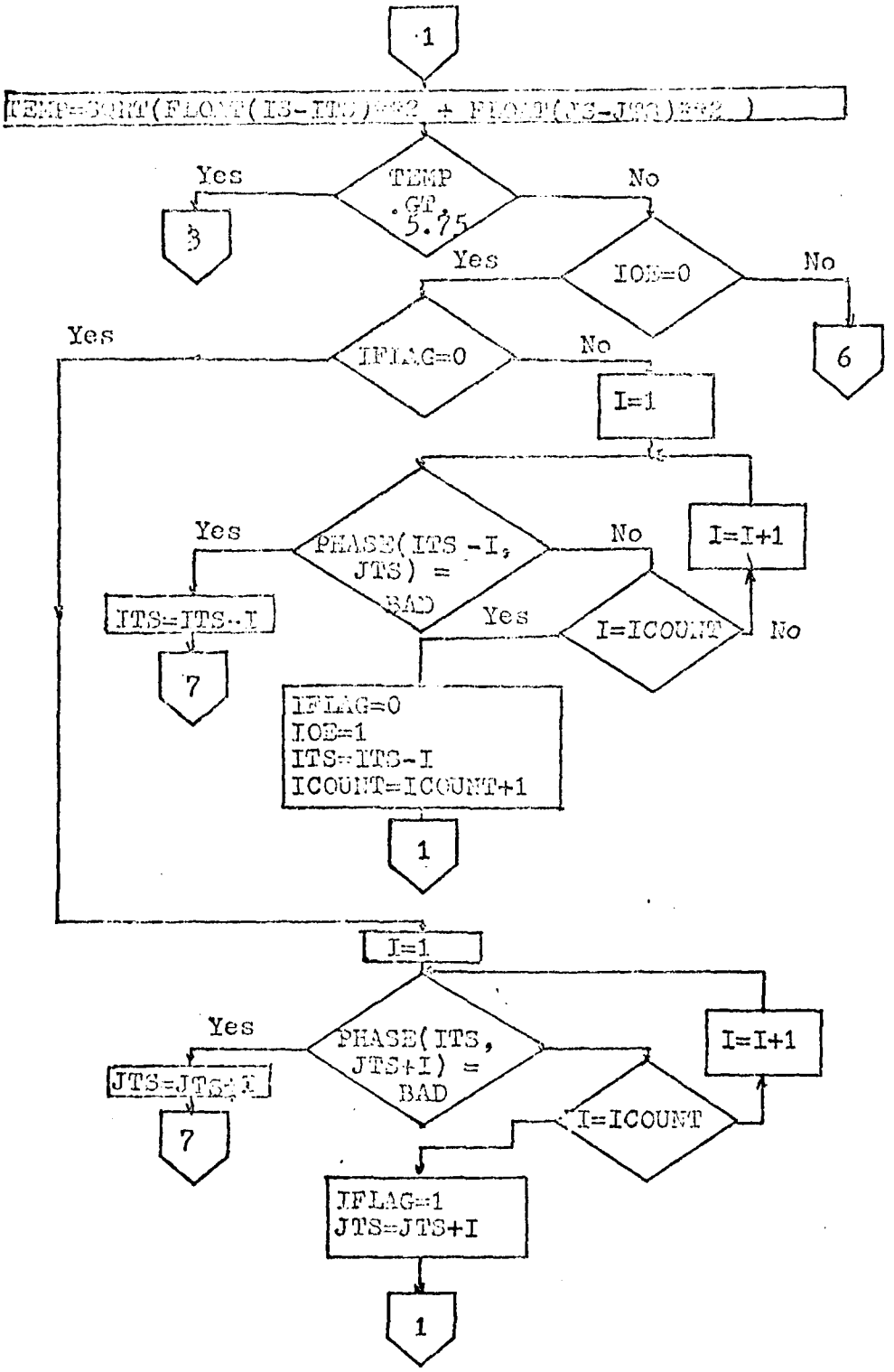


Figure 39-4. Flowchart of FAPER

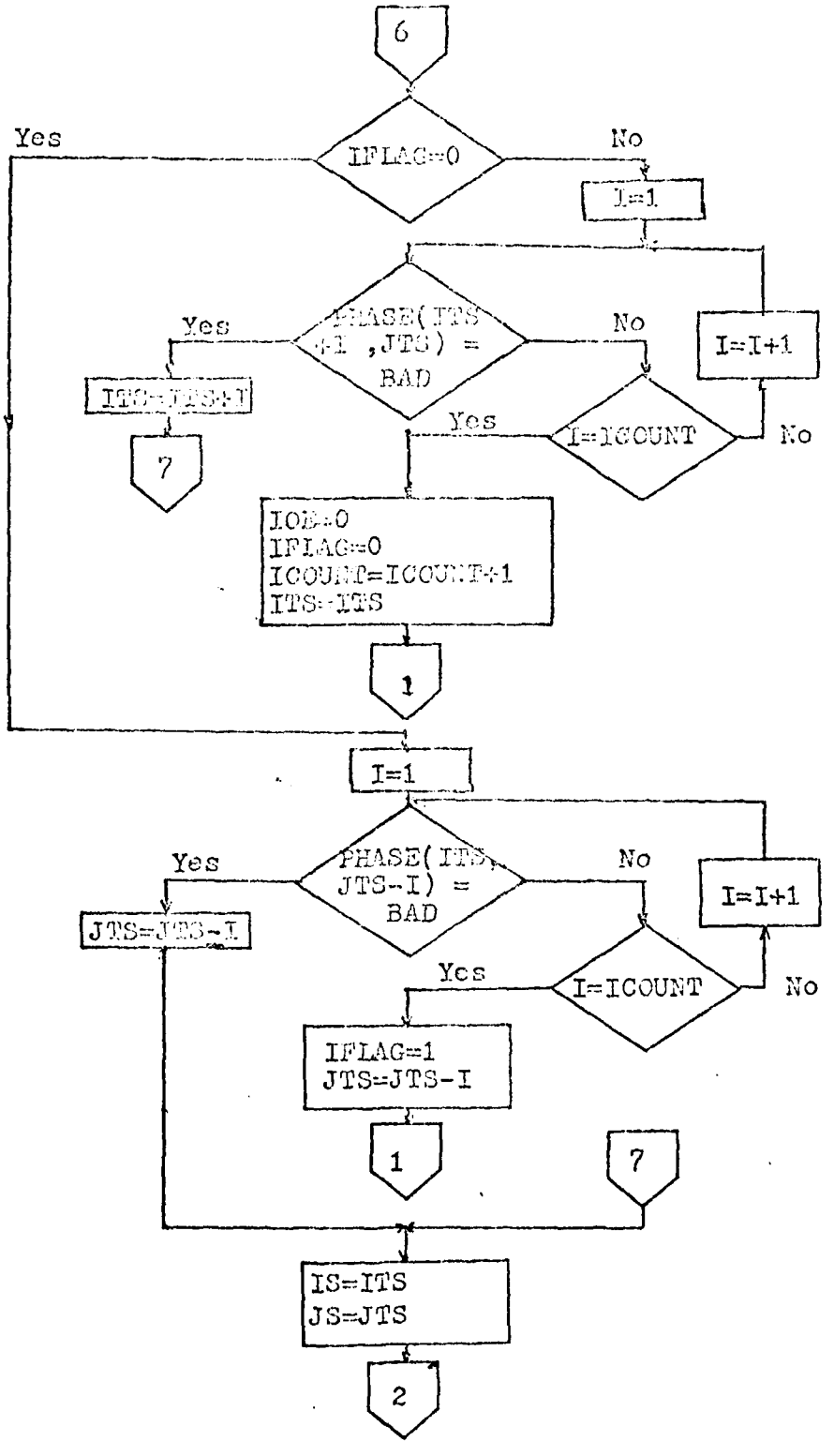


Figure 39-5. Flowchart of FAPER

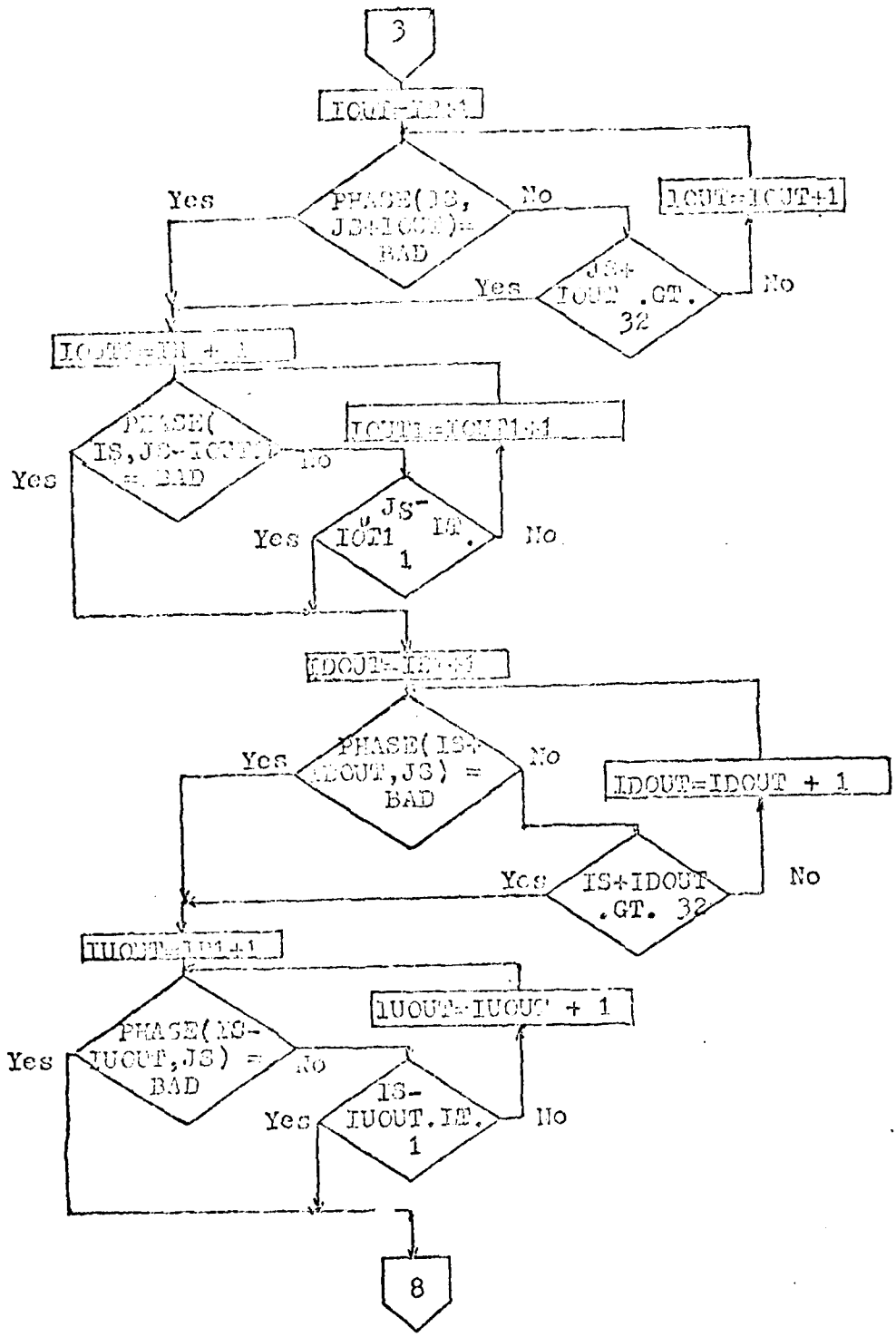


Figure 39-6. Flowchart of FAPER

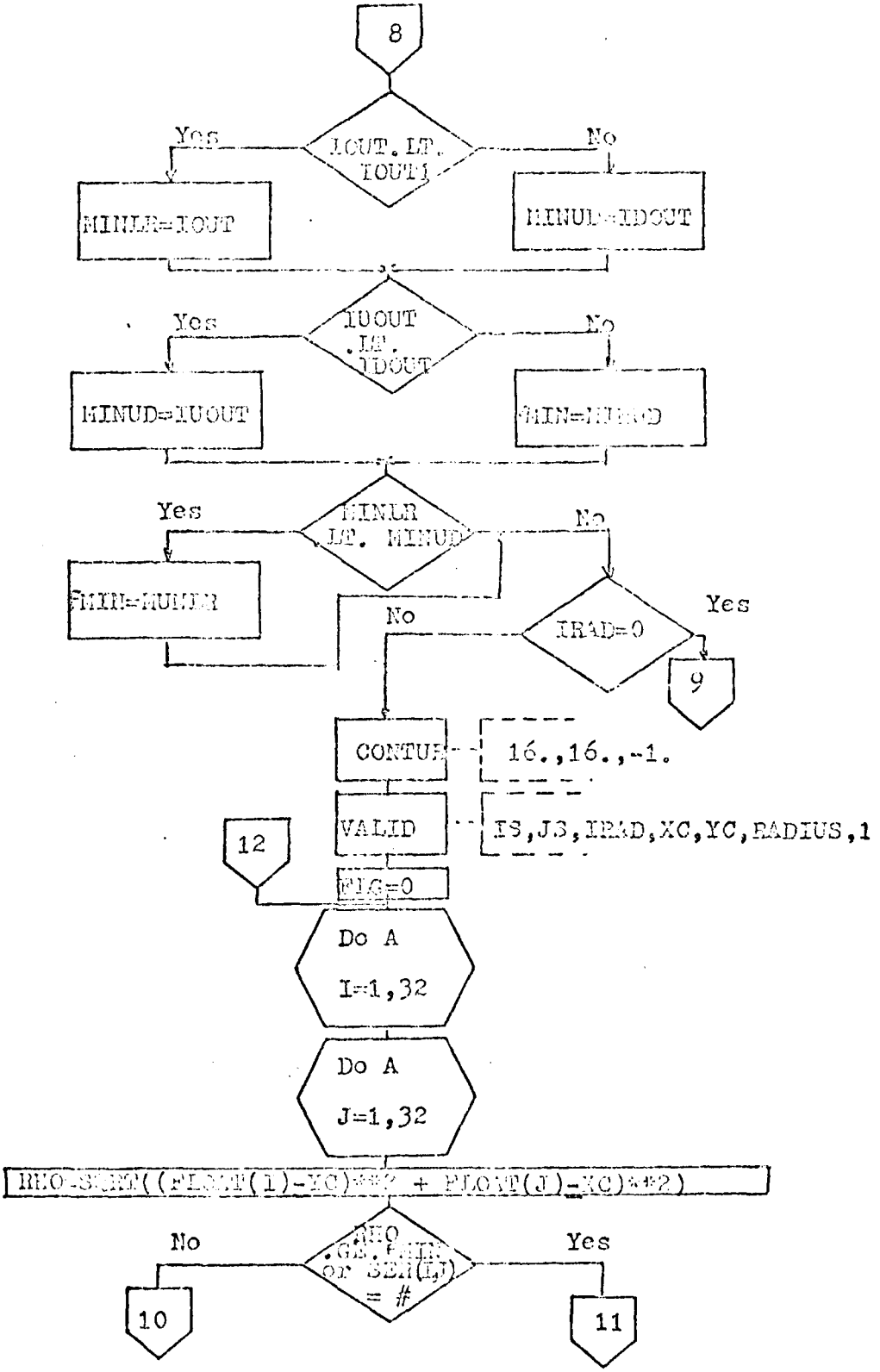


Figure 39-7. Flowchart of FAPER

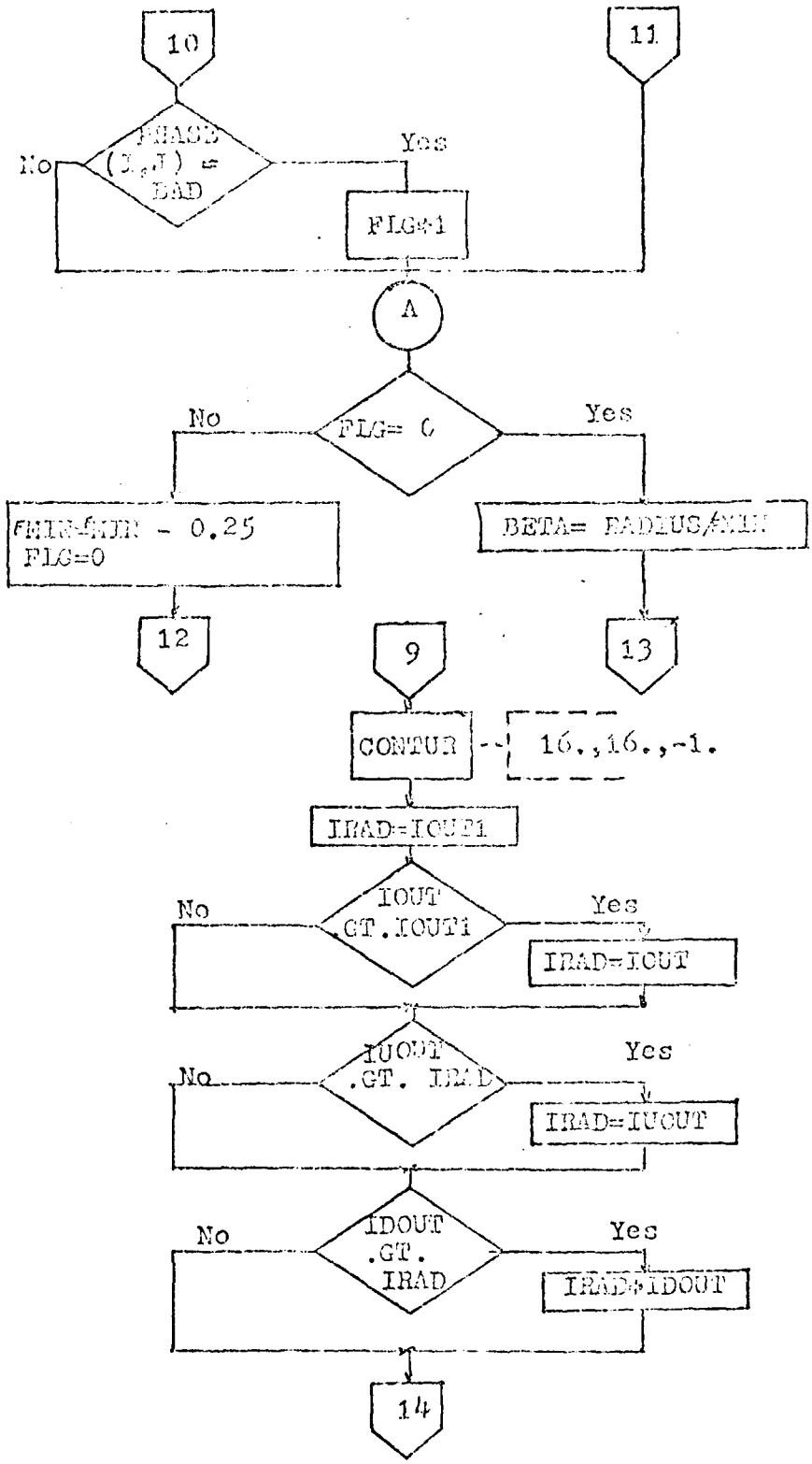


Figure 39-8. Flowchart of PAPER

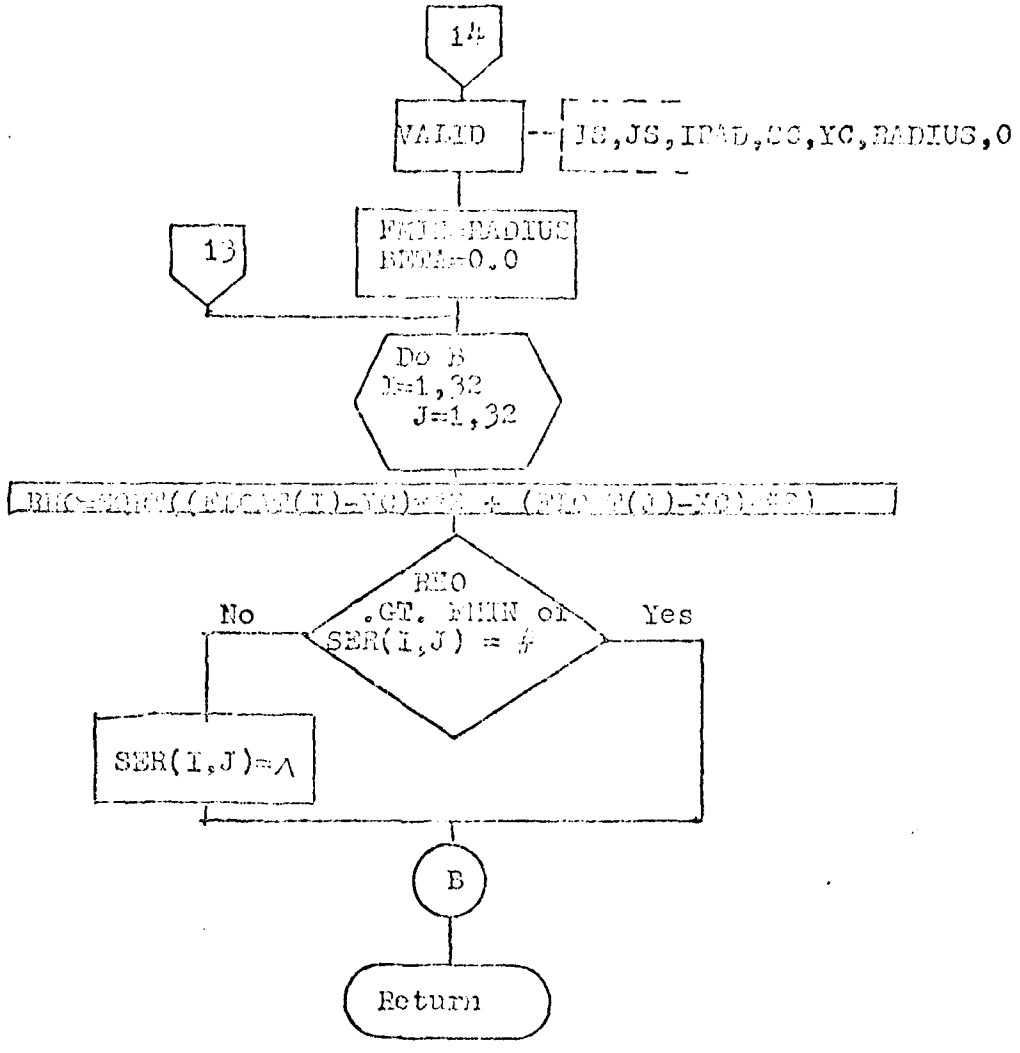


Figure 39-9. Flowchart of FAPER

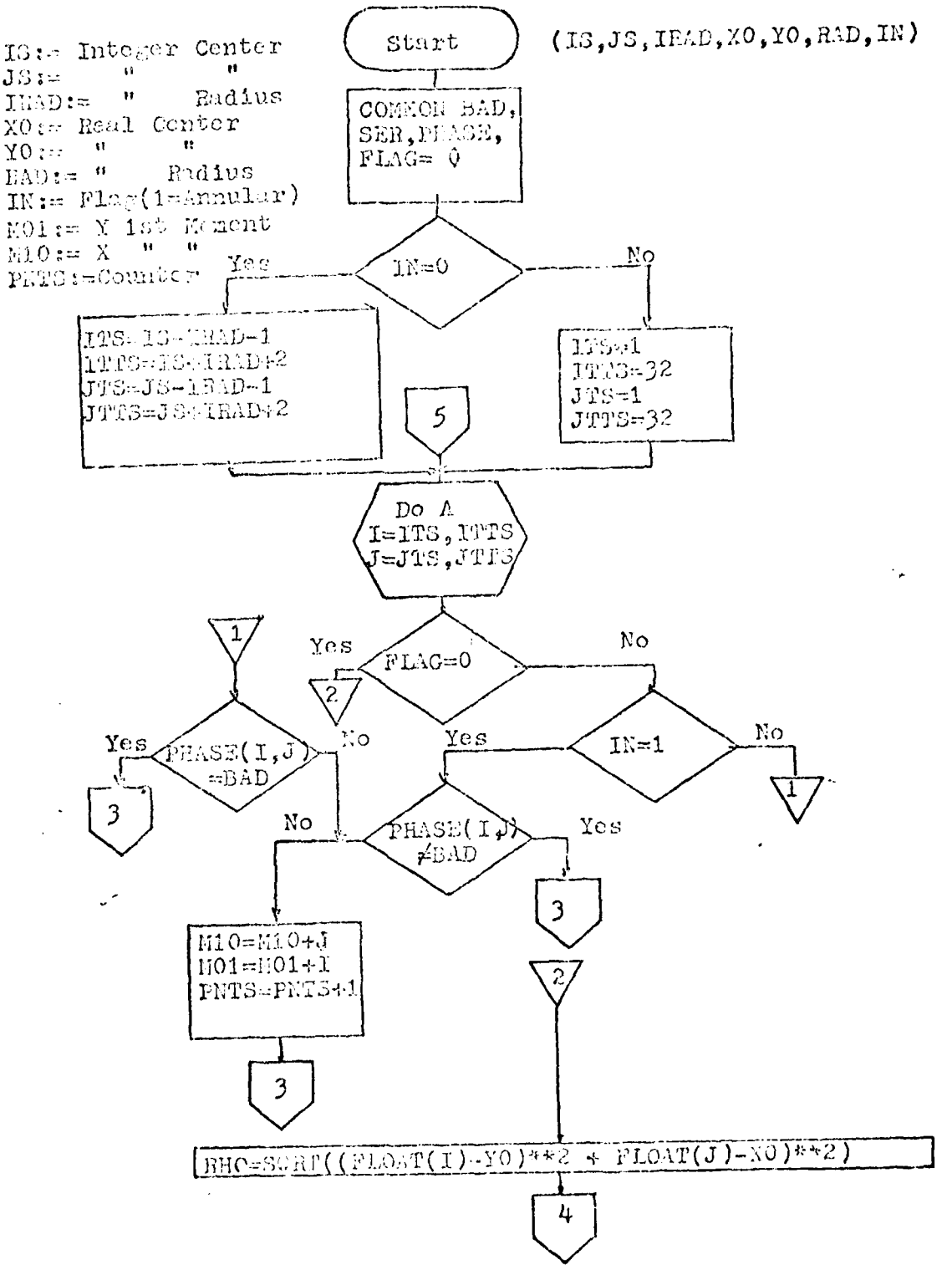


Figure 40-1. Flowchart of VALID

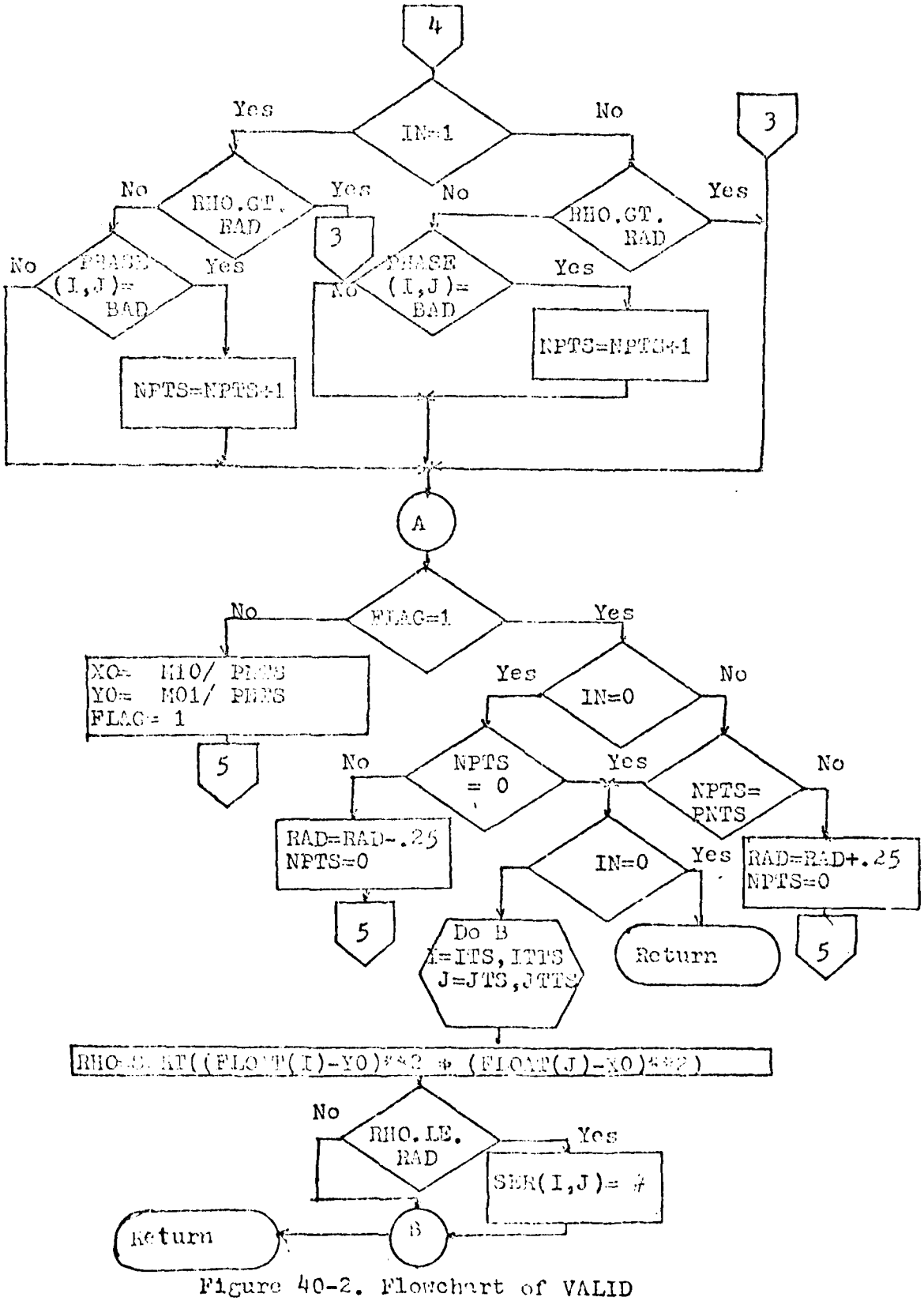


Figure 40-2. Flowchart of VALID

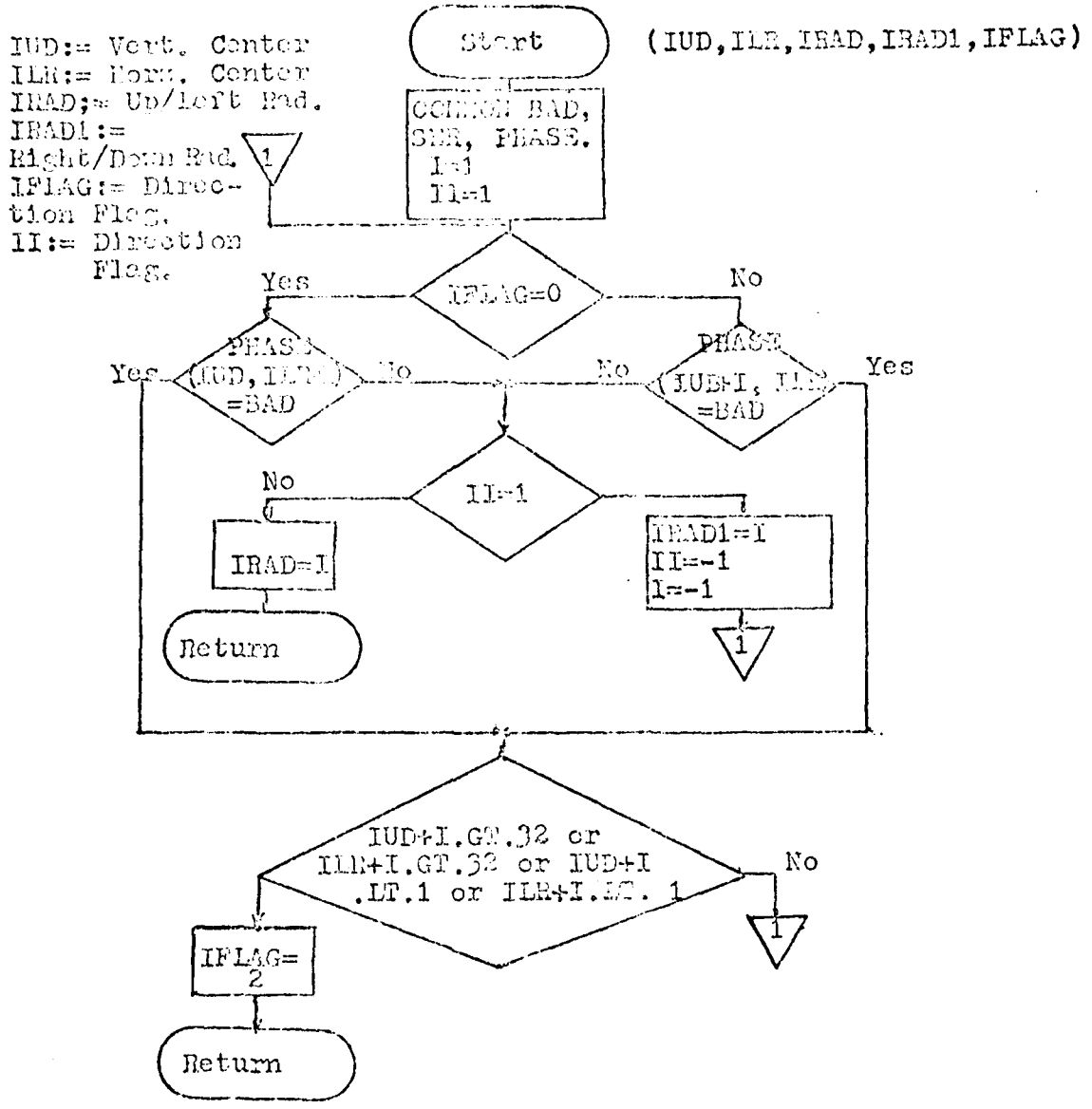


Figure 41. Flowchart of FDGE

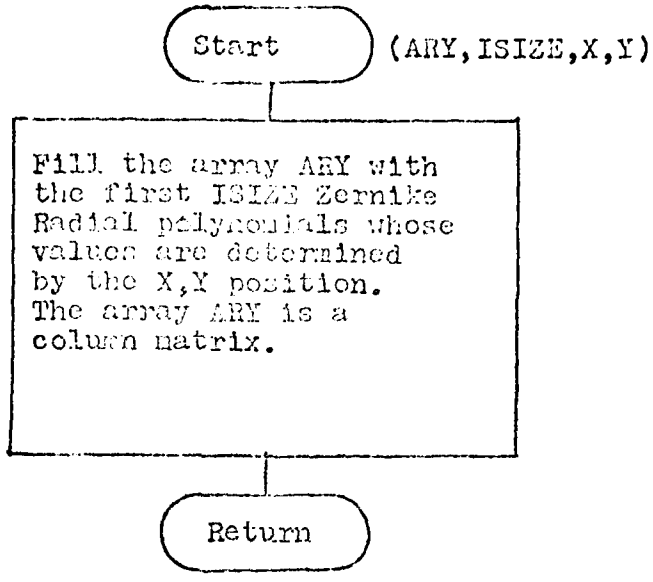


Figure 42. Flowchat of ZRAD

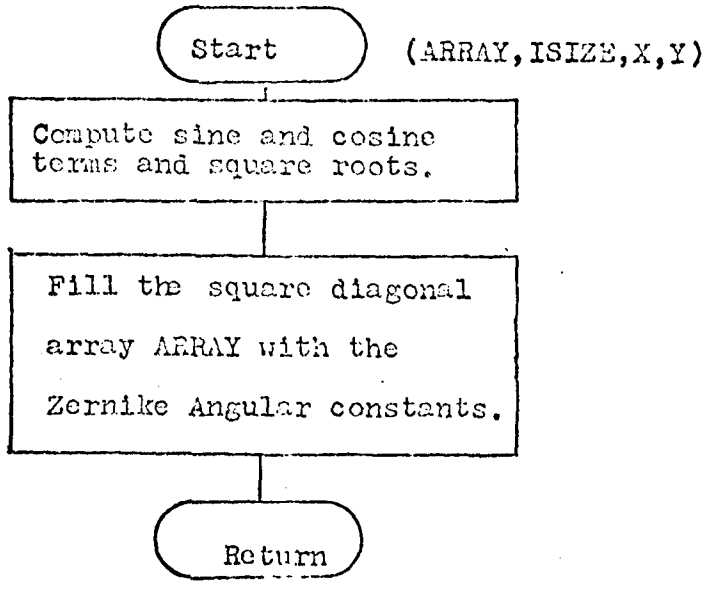


Figure 43. Flowchart of ZANG

XC:= X Center
YC:= Y "
RAD:= Inside Radius

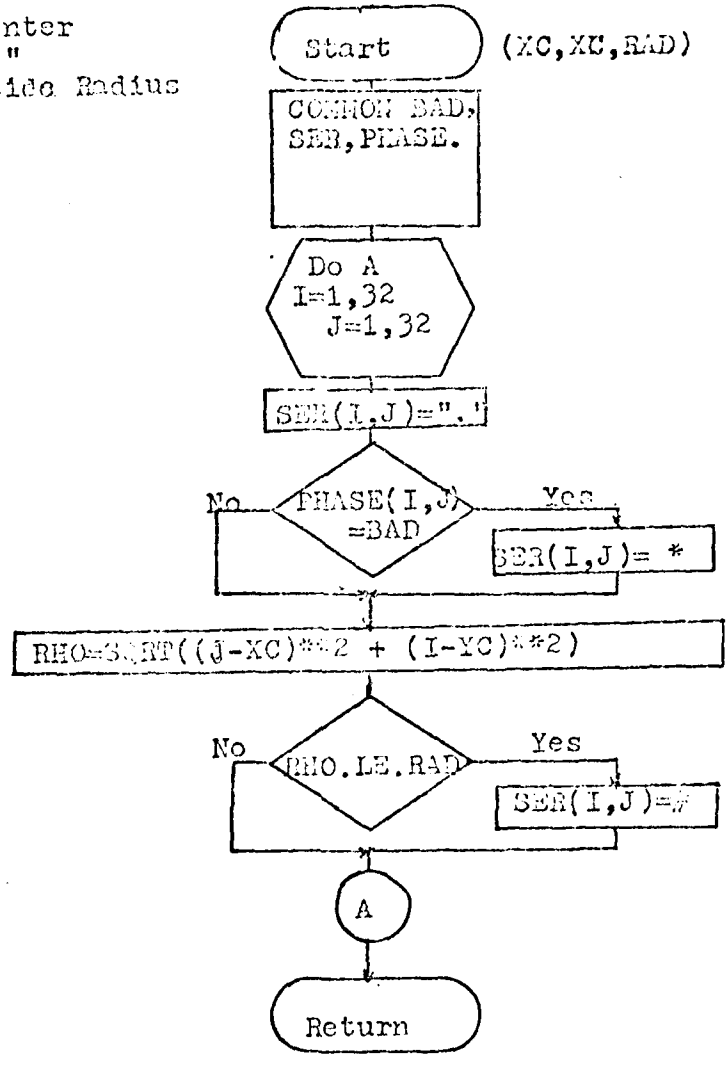


Figure 44. Flowchart of CONTUR

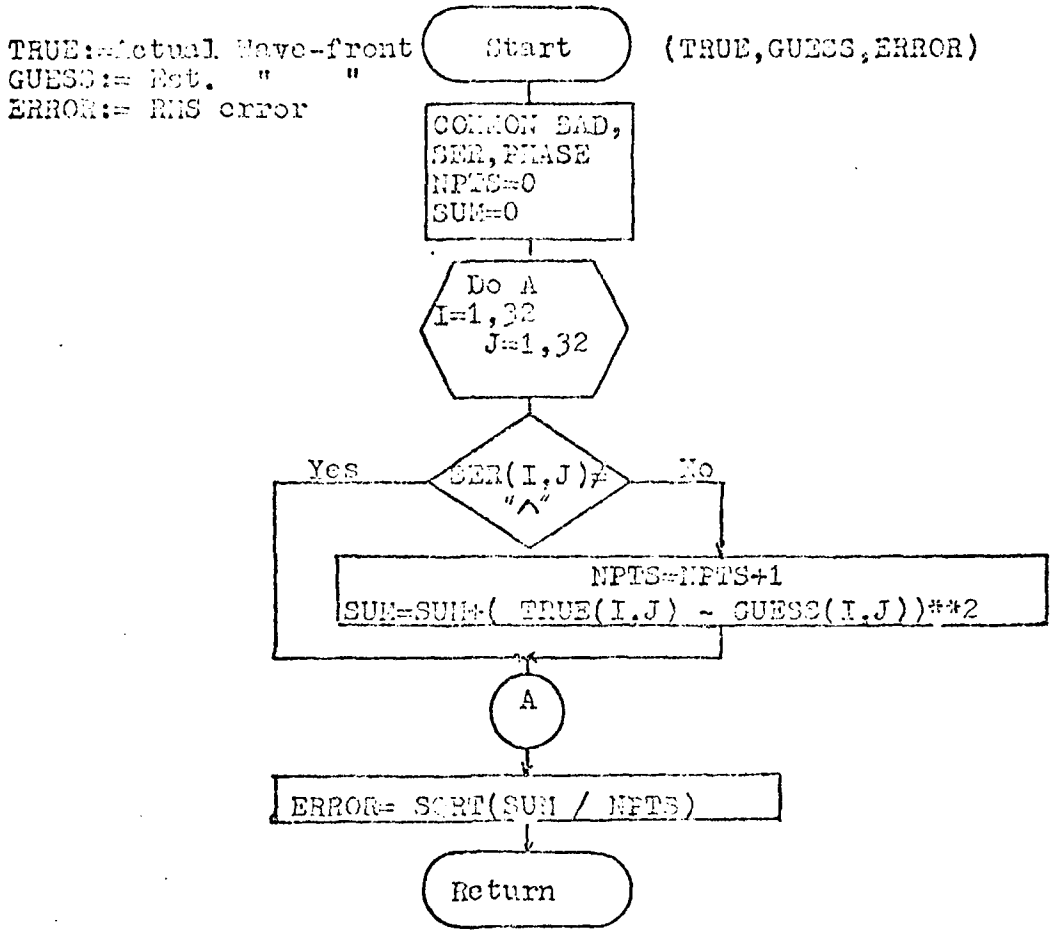


Figure 45. Flowchart of RMSERR

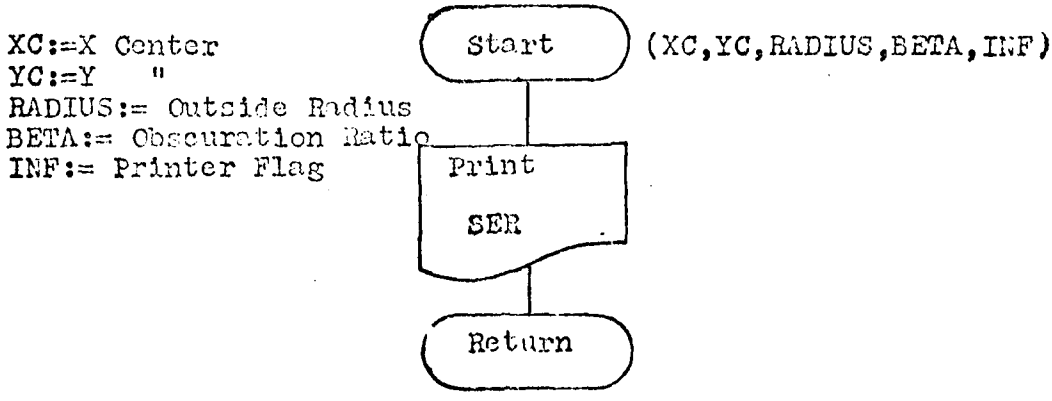


Figure 46. Flowchart of SERPRNT

A:=Input Array
B:=Inverted Array
MAX:= Size of Array

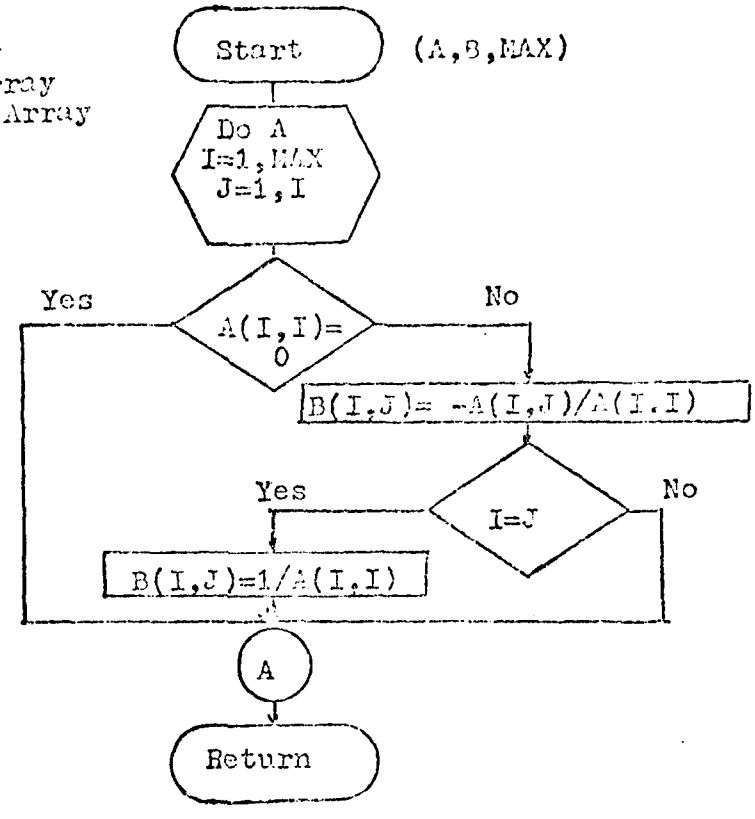


Figure 47. Flowchart of INVERT

A:=Multiplier
B:=Multiplicand
C:=Product
L:= Size of arrays

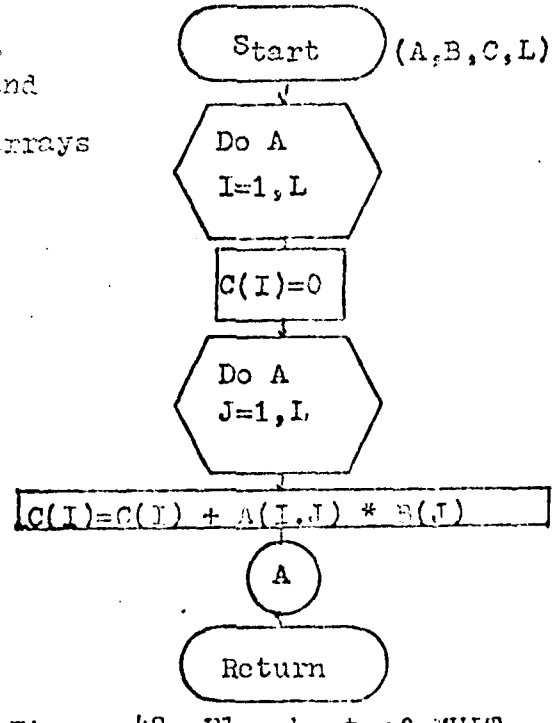


Figure 48. Flowchart of MULT

A:= Array to be printed
ISIZE:= Size
NAME:= Name of array

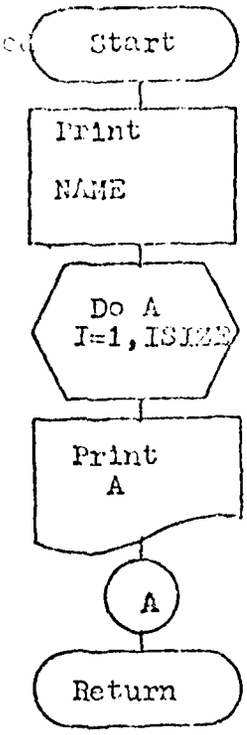


Figure 49. Flowchart of ARPR1

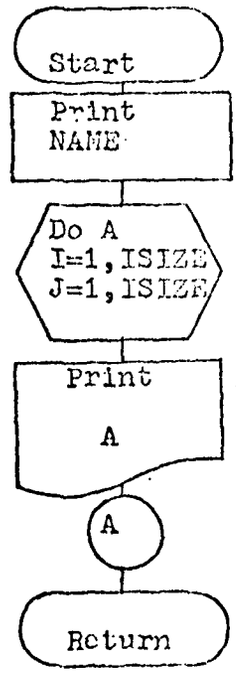


Figure 50. Flowchart of ARPR2

Appendix E

Major Arrays Generated by the Software

Appendix E

Major Arrays Generated by the Software

This appendix contains the arrays generated by the software. The arrays are shown when the software is solving for all 22 coefficients.

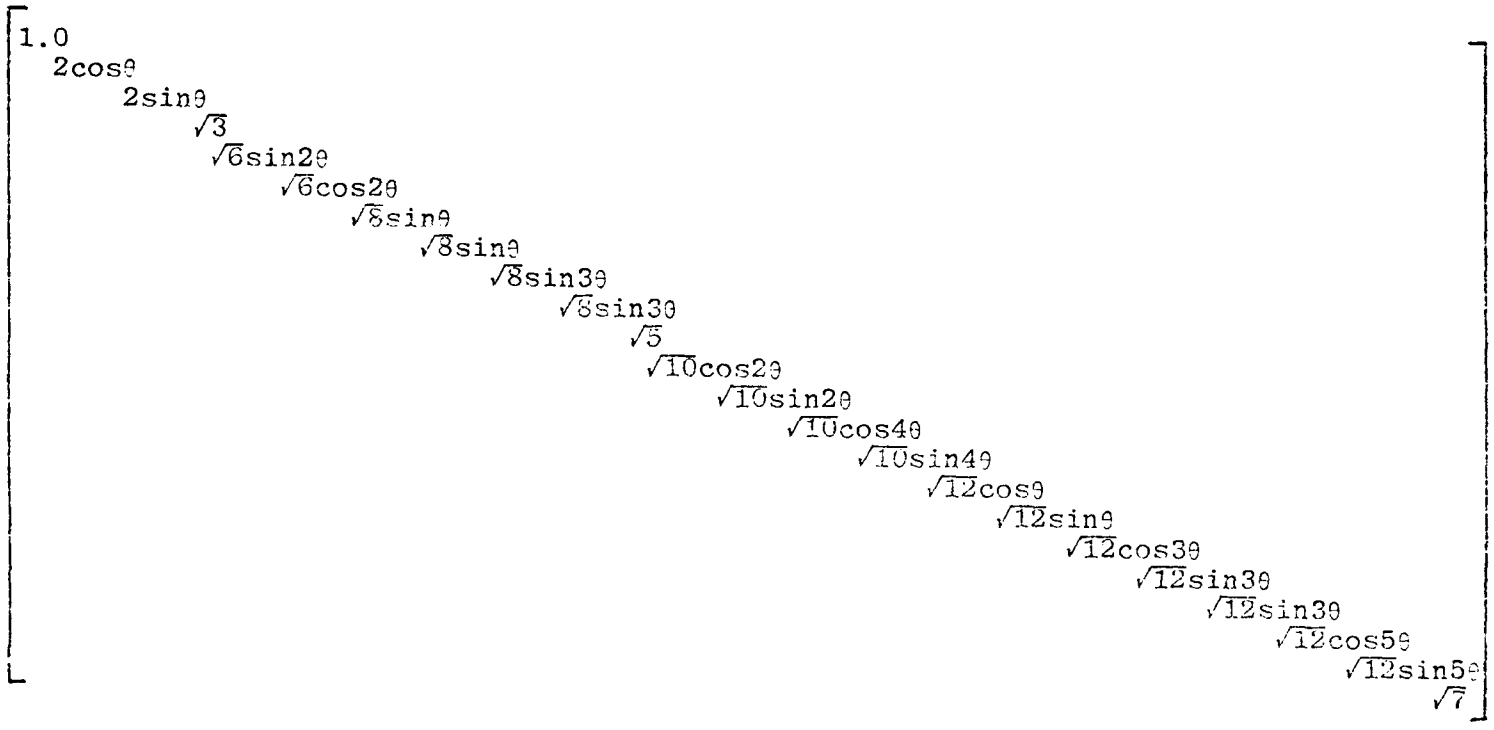
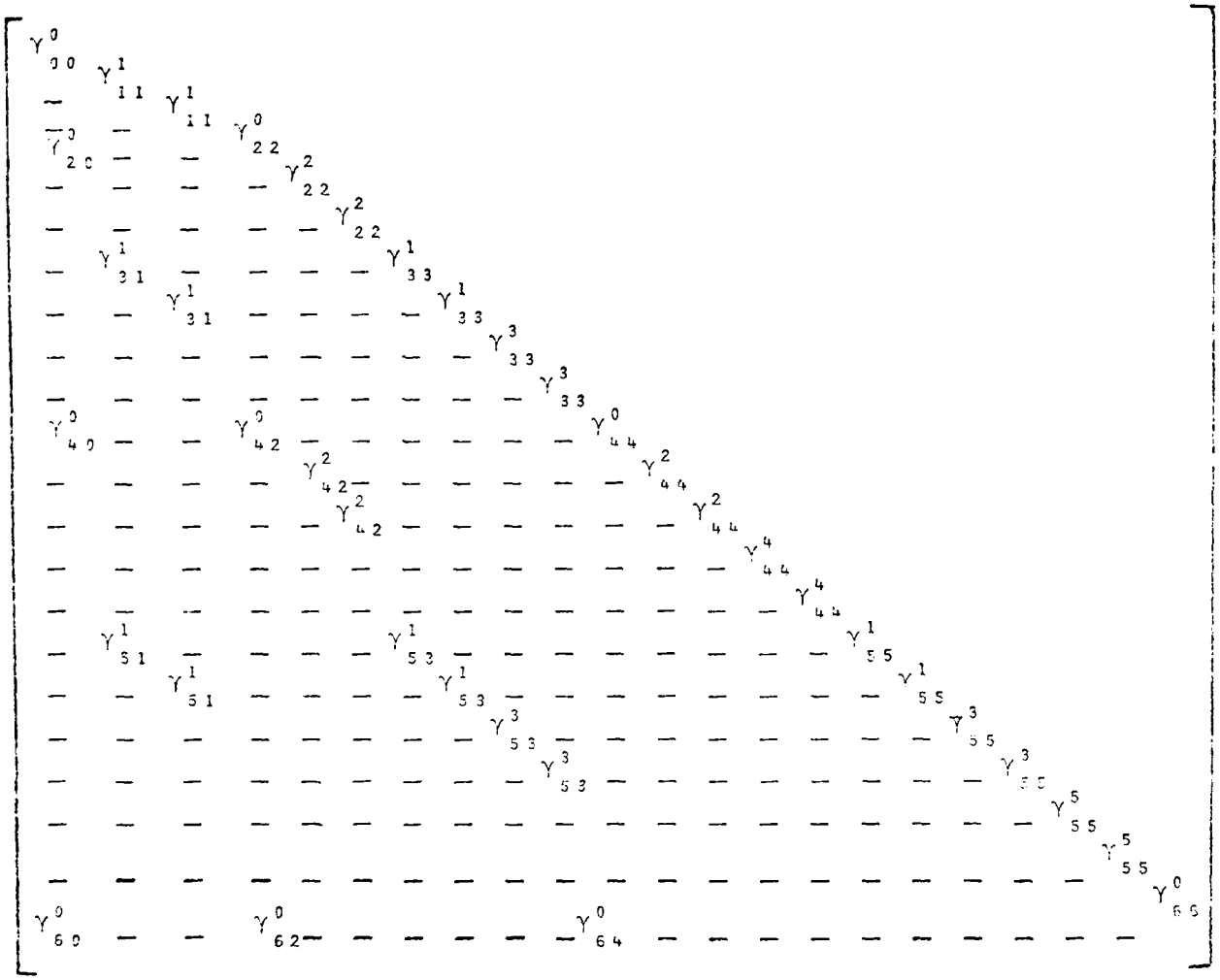


Figure 51. Array Created by ZANG



_ = No Entry

Figure 52. Array Generated by GAMSUB

Appendix F

Program Listing

```
      PROGRAM MAIN(INPUT,OUTPUT,TAPE5,TAPE6=OUTPUT)
C
C     THIS PROGRAM WAS WRITTEN BY LT. ROBERT D. [illegible] AS PART OF
C     A MASTERS DEGREE THESIS FOR THE AIR FORCE INSTITUTE OF TECHNOLOGY.
C     ANY QUESTIONS ABOUT THIS PROGRAM CAN BE SENT
C     TO HIM AT HIS HOME ADDRESS WHICH IS:
C     [illegible] DR.
C     [illegible]FIELD, MASS.
C
C
C        THIS PROGRAM WILL FIND THE COEFFICIENTS OF THE ZERNIKE
C     POLYNOMIALS [illegible] BY A SET OF NEW POLYS WHICH ARE ORTHO-
C     NORMAL OVER THE ANNULUS. THIS PROGRAM IS BASED ON THE PAPER
C     "WAVE-FRONT INTERPRETATION WITH ZERNIKE POLYNOMIALS" BY
C     [illegible] 1980.
C
C
C        THE BASIC ALGORITHM IS:
C
C     1. DETERMINE THE BASIC PARAMETERS OF THE WAVE-FRONT
C     BY [illegible], LET THE USER DECIDE IF HE WANTS A
C     PRINTOUT OF THE [illegible].
C
C     2. DEFINE THE REGION TO BE ANALYSED BY CALLING CONTUR.
C
C     3. ZERO OUT ALL DATA ARRAYS
C
C     4. COMPUTE [illegible] WHICH GENERATES THE W POLYNOMIAL FROM
C     THE ZERNIKE POLYNOMIALS BY CALLING GAMSUP.
C
C     5. COMPUTE THE X AND Y STARTING POSITIONS AND INCREMENTS.
C
C     6. USING THE VALID REGION DEFINED BY CONTUR, COMPUTE THE
C     NEW POLYNOMIALS COEFFICIENTS.
C
C     7. CONVERT NEW COEFFICIENTS BACK TO ZERNIKE COEFFICIENTS.
C
C     8. GENERATE TWO WAVE-FRONTS USING THE ZERNIKE AND NEW
C     COEFFICIENTS.
C
C     9. COMPUTE RMS ERROR AND PRINT OUT THE RESULTS.
C
C        THE PROGRAM WILL LOOP BETWEEN STEPS 2 AND 9 FOR EACH
```

```
C     FRAME OF DATA. STEPS 3 TO 9 ARE DONE THREE TIMES FOR
C     EACH NEW NUMBER OF COEFFICIENTS, I.E. [illegible],11, [illegible].
C
C     DEFINITION OF VARIABLES
C
C     ARRAYS
C
C     PHASE      ACTUAL PHASE-FRONT DATA
C     PHASE1     REFITTED PHASE-FRONT USING ZERNIKE POLYNOMIAL [illegible]
C     PHASE2        "        "      "     "  [illegible]      "      "
C     [illegible] DEFINES NO DATA, VALID DATA, DATA [illegible], AND [illegible]
C     SIGMA      GIVES ZERNIKE RADIAL TERMS [illegible]
C                TERMS
C     INGAMMA    INVERSE OF GAMMA
C     [illegible] [illegible] PORTION OF ZERNIKE POLYNOMIALS.
C     [illegible] [illegible] OF [illegible]
C     [illegible] [illegible] COEFFICIENTS
C     [illegible] [illegible] COEFFICIENTS
C     [illegible] [illegible] ZERNIKE POLYNOMIALS
C     [illegible] [illegible] POLYNOMIALS
C     [illegible] [illegible] NEW POLYNOMIALS
C     [illegible] ZERNIKE POLYNOMIAL VALUES
C     [illegible] [illegible]
C
C     NON-ARRAY VARIABLES
C
C     [illegible] VALUE USED TO DEFINE NO DATA POINTS
C     [illegible] OBSCURATION RATIO
C     [illegible] [illegible]
C     [illegible] [illegible]
C     [illegible] DUMMY VARIABLE USED TO SKIP INTENSITY ARRAY FROM [illegible]
C     RADIUS     [illegible] OF WAVEFRONT
C     X0         X AXIS CENTER OF DEFINED REGION OF WAVEFRONT
C     Y0         Y  "     "     "    "      "    "     "
C     NY         NO/YES  USED IN INQUIRY SECTION
C     [illegible] INSIDE RADIUS OF ANNULUS
C     NFRMS      NUMBER OF FRAMES TO BE READ
C     [illegible] NUMBER OF COEFFICIENTS TO BE FOUND [illegible]
C     [illegible] X [illegible]
C     [illegible] Y    "     "     "     "    "     "
C     X ,Y       COORDINATE OF STARTING POINT FOR ANALYSIS
C     Y,Y            "      "  CURRENT POSITION [illegible]
C     I,J        INDEX INTO DATA ARRAY WHICH CORRESPONDS TO Y,Y
C     [illegible] TOTAL NUMBER OF VALID [illegible]
C     [illegible] [illegible]
C     [illegible] CURRENT NUMBER OF COEFFICIENTS BEING FOUND
C     [illegible],K GENERAL PURPOSE COUNTERS
```

```
SUBROUTINES CALLED (EACH INDENT INDICATES LEVEL OF [illegible])

MAIN
   [illegible] - DEFINES REGION OF VALID DATA
      [illegible] - PRINTS THE [illegible] SER
      CONTUR
         SORT
      EDGE - FINDS INSIDE EDGE OF ANNULUS
      VALID - FIND ACTUAL CENTER AND [illegible]
         SORT
      SORT
   CONNUM - FILLS SER WITH PROPER SYMBOLS
      SORT
   GAMOUT - COMPUTES GAMMA
      SORT
   INVTRI - INVERTS LOWER TRIANGULAR MATRIX
   ZRAD - COMPUTES ZERNIKE RADIAL TERMS
      SORT
   ZANG - COMPUTES ANGULAR TERMS FOR ZERNIKE POLYNOMIALS
      SORT
   MULT - MULTIPLIES A (M*N) ARRAY TIMES A (N) ARRAY
   RMSERR - COMPUTES RMS ERROR BETWEEN TWO ARRAYS
      SORT
   SERPRNT - PRINT OUT ARRAY SER

FROM BASIC ALGORITHM THE FOLLOWING ARRAYS [illegible]
STEPS 6 TO 8

STEP 6

!C1! !1                 ! !1/G111                               ! [illegible]
!C2!=!  2COS            ! !           1/G111                    ! [illegible]
!C3! !      2SIN        ! !                  1/G111             ! [illegible]
!C4! !            SQRT3 ! !-G [illegible]/G122           1/G [illegible] ! [illegible]

SPOLY=  SCNST             [illegible]       [illegible]              [illegible]
SPOLY=  SCNST             [illegible]       NEW RADIAL TERMS

NEWCOEF(I)=(SUM(PHASE*SPOLY(I)) OVER ALL VALID DATA)/TOTAL

STEP 7
!T1! !1                 ! !G[illegible]                ! !                    ! [illegible]
!T2! !  2COS            ! ![illegible]     G111        ! !  1/2COS            ! [illegible]
!T3!=!      2SIN        ! ![illegible]       G111      ! !        1/2SIN      ! [illegible]
```

```
C    !ZN! !?          SQRT3! ! 0        0      3122! !                    1/SQRT3! !NC !
C
C    ZCOEF=  COUST          *        GAMMA                       INCONST
C    Z COEF=  COUST         .        GAMMA
C    ZCOEF=  COUST          +        ZERNIKE RADIAL TERMS COEFFICIENTS
C
C        ZCOEF(I)=(SUM((ZTEMP) OVER ALL VALID DATA)/TOTAL
C
C
C        STEP 8
C
C                 !ZC1! !1                           ! !R1!
C    CPHASE1=     !ZC2! !   2COS                     ! !R2! EVALUATED AT EACH VALID X,Y
C                 !ZC3! !          2SIN              ! !R3!
C                 !ZC4! !                SQRT3!        !R4!
C
C                 !NC1! !1                           ! !1/3.??                          ! !??!
C    CPHASE2=     !NC2! !   2COS                     ! !            1/3134              ! !??!
C                 !NC3! !          2SIN              ! !                      1/3444    ! !??!
C                 !NC4! !                SQRT3!        !-3.21/3.22                 ??! !??!
C
C

      COMMON PHASE(32,32),SEP(32,32),RAD
      REAL INGAMMA(22,22),NEWCOEF(22)
      REAL ZPOLY(22)
      DIMENSION GAMMA(22,22),ZRADIAL(32)
      DIMENSION CPHASE1(32,32),ZCOEF(22),CPHASE2(32,32)
      DIMENSION COUST(22,22),INCONST(22,22),GRAD(22),?POLY(22)
      DIMENSION ZT??F(22)
      INTEGER FR?
      ?NT?D E

C
C
C        THE FIRST FRAME OF DATA IS READ IN TO HELP THE
C         THE USER GET AN IDEA OF WHAT THIS PROGRAM WILL BE
C         USING FOR DATA.  F?PEP IS CALLED TO FIND WHAT IT
C         "THINKS" THE CENTER ,RADIUS, AND ?????????? RATIO
C        SHOULD BE. THE USER IS ASKED IF HE/SHE WANTS TO CHANGE
C         THE DATA.  THIS IS DONE BECAUSE A HEMISPHERICAL ?????
C         WAVE-FRONT WOULD INTRODUCE AN ERROR IF THE ???? ?????
C         WAS A CIRCULAR WAVE-FRONT OBSCURED BY AN ?????-????.
```

```
C
C
C
      READ(5)XRN,FRM,((PHASE(I,J),J=1,32),I=1,32),
     *((B,I=1,32),J=1,32)
      CALL FARF.(XC,YC,RADIUS,BETA)
C     PRINT OUT THE BASIC INPUT FRAME
      WRITE (6,1)
1     FORMAT(" ",5X,"ENTER 1 IF YOU WANT TO SEE THE FIRST FRAME ")
      READ ,NY
      PRINT ,NY
      IF(NY .EQ. 1)CALL SERPRNT(XRN,FRM,XC,YC,RADIUS,BETA, )
C     THIS SECTION WILL READ IN ANY INPUT DATA. THE PRINT [illegible]
C     USED TO ECHO PRINT THE INPUT VARIABLES IF THIS ROUTINE IS
C     RUN IN A BATCH MODE INSTEAD OF INTERACTIVE.
C     THESE [illegible] CHANGED IF THE OPERATING SYSTEM DOES [illegible]
C     FREE FORMAT PRINT AND READ STATEMENTS. [illegible]
C     I/O OPERATION ON THE DEFAULT INPUT/OUTPUT PORTS.
C
C
      WRITE(6,2)
2     FORMAT(" ",5X,"ENTER THE NUMBER OF FRAMES TO BE READ IN ")
      READ ,NFRMS
      PRINT ,NFRMS
      WRITE (6,3)
3     FORMAT(" ",5X,"USER DEFINED REGION? (1=YES, =NO)    ")
      READ ,NY
      PRINT ,NY
      IF(NY .NE. 1)GO TO 9
      WRITE(6,4)
4     FORMAT(" ",5X,"ENTER X AND Y FOR USER DEFINED CENTER",//,5X,
     *"(MAKE X=-1 IF YOU AGREE WITH THE PROGRAM'S CENTER)")
      TEMP= XC
      READ ,XC
      IF(XC .EQ. -1)GO TO 5
      READ ,YC
      PRINT ,XC," ",YC
      GO TO  
5     XC= TEMP
      WRITE (6,7)
7     FORMAT(" ",5X,"ENTER RADIUS (SET TO -1. IF  OKAY)")
```

```
      TEMP= RADIUS
      READ(*,*)RADIUS
      PRINT*,RADIUS
      IF(RADIUS .EQ. -1)RADIUS= TEMP
      WRITE(6,3)
3     FORMAT(" ",1X,"ENTER OBSCURATION RATIO (SET TO -1 IF OKAY)")
      TEMP=BETA
      READ(*,*)BETA
      PRINT*,BETA
      IF(BETA .EQ. -1)BETA= TEMP
      RAD= RADIUS * BETA
      CALL CONTUR(XC,YC,RAD)
C
C     THIS IS THE START OF THE MAJOR LOOP. THE UPPER LIMIT IS THE
C     DESIRED NUMBER OF FRAMES TO BE READ. NOTICE THAT THE FOLLOWING
C     READ IS NOT DONE FOR THE FIRST RUN SINCE IT WAS ALREADY [illegible]
C     READ. THE CALL TO CONTUR IS USED TO FILL THE ARRAY [illegible]
C     THE PROPER SYMBOLS FOR USE LATER ON.
C
C
      DO 3333 NFNI= 1,NFRMS

      IF(NFRMS .EQ. 1)GO TO 1
      READ(1)X,Y,FRN,((PHASE(I,J),J=1,32),I=1,32),
     *((?,I=1,32),J=1,32)

      IF(NFRMS .NE. 1)CALL CONTUR(XC,YC,RADIUS * BETA)
C
C     THIS LOOP IS WHERE THE NUMBER OF COEFFICIENTS TO BE FOUND IS
C     DETERMINED. WHEN NTRMS IS 1, 6 COEFFICIENTS ARE TO BE FOUND
C     2 GIVES 11 COEFFS, AND 3 THEN 22 COEFFICIENTS.
C
1     DO 3333 NTRMS=1,3

      IF(NTRMS .EQ. 1)NCOEF= 6
      IF(NTRMS .EQ. 2)NCOEF= 11
      IF(NTRMS .EQ. 3)NCOEF= 22

C
C     THIS IS STEP 3 OF THE ALGORITHM: IT CLEARS OUT ALL
C     OF THE DATA ARRAYS FOR THE NEXT RUN.
C
```

```
      DO 30  I=1,32
         DO 20  J=1,32
            CPHASE0(I, J)=0.0
20          CPHASE1(I, J)=0.0
30    CONTINUE
      DO 25  I= 1,NCOEF
         ZERCOEF(I)=0.0
         ZCOEF(I)= 0.0
         SPOLY(I)=0.
         SZER(I)=0.
         ZTEMP(I)=0.
         ZPOLY(I)=0.
         DO 35  J= 1,NCOEF
            GAMMA(I,J)= 0.0
            INGAMMA(I,J)=0.0
            COUNT(I,J)=0.
            [illegible]COUNT(I,J)=0.0
35       CONTINUE
25    CONTINUE
C
C     THIS IS STEP 4 OF THE ALGORITHM.  AT THIS POINT
C     THE TRANSFORMATION COEFFICIENTS ARE CALCULATED BY
C     CALLING GAMSUB. INVERT IS CALLED TO ALLOW THE
C     PROGRAM TO EASILY CONVERT ZERNIKE COEFFICIENTS TO
C     NEW POLY COEFFICIENTS, AND VISA VERSA.
C

      CALL  GAMSUB(GAMMA,NCOEF,BETA)
      CALL  INVERT(GAMMA,INGAMMA,NCOEF)

C     THE FOLLOWING CODE IS STEP 5 IN THE ALGORITHM. USING THE
C     VALUES EITHER ENTERED OR COMPUTED BY FIRST, THE LIMITS
C     FOR THE QUADRANT II (X0,Y0) ARE COMPUTED. USING THE
C     RECIPROCAL OF THE RADIUS, THE INCREMENTS IN EACH
C     OF THE CARTESIAN COORDINATES ARE COMPUTED (XINC,YINC).
C

      XINC= 1.0/RADIUS
      YINC= -1.0/RADIUS
      Y[illegible]= (1.0 - YO)/RADIUS
      Y[illegible]= (YO - 1.0)/RADIUS
      X= X[illegible]
      Y= Y[illegible]
```

```
C
C     THE FOLLOWING CODE COMPUTES THE NEW POLYNOMIAL COEFFICIENTS
C     OF STEP [illegible] OF THE ALGORITHM. AS POINTED OUT [illegible], THE
C     PROPER ARRAYS ARE CONSTRUCTED AT EACH X AND Y [illegible], AND
C     THE COEFFICIENTS FOR THAT POINT ARE CALCULATED [illegible]
C     SUM OF EACH COEFFICIENT IS KEPT. AFTER [illegible]
C     PHASE, THESE SUMS ARE DIVIDED BY THE NUMBER OF POINTS
C     USED TO CALCULATE THE COEFFICIENTS.
C
      TOTAL= [illegible]
      DO [illegible] I= 1,32
         Y =Y[illegible]
         DO 3[illegible] J= 1,32
            IF(S[illegible](I,J) .EQ. 1H[illegible] .OR. SER(I,J) [illegible]) GO TO 2[illegible]
            IF(SQRT(X**2 + Y**2) .GT. 1. )GO TO 2
            CALL Z[illegible](Z[illegible]DIAL,NCOEF,X,Y)
            SE[illegible](I,J) = 1H[illegible]
            CALL [illegible](CO[illegible]ST,NCOEF,X,Y)
            CALL MULT([illegible],Z[illegible],S[illegible]AD,NCOEF)
            CALL MULT(CONST,SRAD,SPOLY,NCOEF)
            TOTAL=TOTAL + 1.
            DO [illegible] K= 1,NCOEF
               NEWCOEF(K)=NEWCOEF(K) + PHASE(I,J) [illegible]SPOLY(K)
1[illegible]        CONTINUE
2[illegible]        X= X + XINC
3[illegible]     CONTINUE
         Y= Y + YINC
4[illegible]  CONTINUE
      DO [illegible] I= 1,NCOEF
45[illegible]    NEWCOEF(I)= NEWCOEF(I) / TOTAL
C
C     THIS IS THE START OF STEP 7 OF THE ALGORITHM. AT THIS POINT
C     THE NEW POLYNOMIAL COEFFICIENTS HAVE BEEN COMPUTED AND
C     ARE READY TO BE CONVERTED TO ZERNIKE COEFFICIENTS.
C     THE PROCESS, AS DETAILED ABOVE, CONVERTS THE NEW POLYNOMIAL
C     COEFFICIENTS TO NEW RADIAL COEFFICIENTS TO ZERNIKE [illegible]
C     COEFFICIENTS TO ZERNIKE POLYNOMIAL COEFFICIENTS. THIS
C     IS DONE AT EACH POINT IN THE ARRAY AND THE RUNNING
C     TOTALS ARE DIVIDED BY THE NUMBER OF POINTS USED.
C
```

```
      TOTAL = [illegible].

      Y=Y[illegible]
      DO 4[illegible] I= 1,32
         X= X[illegible]
         DO 3[illegible] J= 1,32
            IF(SER(I,J) .NE. 1H^)GO TO 2[illegible]
            TOTAL= TOTAL + 1.
            CALL ZANG(CONST,NCOEF,X,Y)
            CALL INVERT(CONST,INCONST,NCOEF)
            CALL MULT(INCONST,INCOEF,SRAD,NCOEF)
            CALL MULT(S[illegible],SRAD,ZRADIAL,NCOEF)
            CALL MULT(CONST,ZRADIAL,ZPOLY,NCOEF)
            DO 1[illegible] K= 1,NCOEF
1[illegible]           ZCOEF(K)= ZCOEF(K)+ZPOLY(K)
2[illegible]        Y=Y+YINC
3[illegible]     CONTINUE
         Y= Y +YINC
4[illegible]  CONTINUE

      DO 5[illegible] I= 1,NCOEF
5[illegible]     ZCOEF(I)= ZCOEF(I) /TOTAL

C
C     AT THIS POINT THE COEFFICIENTS FOR THE TWO POLYNOMIALS HAVE
C     BEEN COMPUTED. STEP [illegible] IS STARTED. THIS STEP COMPUTES THE
C     ESTIMATED PHASE-FRONTS. THE WAVE-FRONT USING THE Z[illegible]
C     COEFFICIENTS IS COMPUTED WITHIN THE [illegible]
C     [illegible] OF THE NEW COEFFICIENTS, THE REGION IS THE [illegible]
C     THE OUTER RADIUS AND THE INSIDE RADIUS.
C

      Y=Y[illegible]
      DO 3[illegible] I= 1,32
         X= X[illegible]
         DO 2[illegible] J= 1,32
            IF(SQRT(X**2 + Y**2) .LE. 1. ) GO TO [illegible]
            OPHASE1(I,J)= BAD
            OPHASE2(I,J)= BAD
            GO TO [illegible]
4[illegible]        CALL ZRAD(ZRADIAL,NCOEF,X,Y)
            CALL ZANG(CONST,NCOEF,X,Y)
            DO [illegible] K= 1,NCOEF
4[illegible]           ZPOLY(K)= ZRADIAL(K) * ZCOEF(K)
            CALL MULT(CONST,ZPOLY,ZTEMP,NCOEF)
```

```
          DO 3[illegible] K= 1,NCOEF
[illegible]          CPHASE1(I,J)= CPHASE1(I,J) + Z[illegible](K)
C
C     THIS SECTION WILL USE THE NEW POLYS TO GEN[illegible] THE W[illegible]-FRONT
C     NOTE THAT BETA IS USED INSTEAD OF RELL [illegible] RADIUS SINCE
C     THE RADIUS (SQRT(X**2 + Y**2)) IS NORMALIZED TO 1.
C
          IF(SQRT(X**2 + Y**2) .GE. BETA)GO TO [illegible]
          CPHASE2(I,J)=BAD
          GO TO [illegible]
[illegible]       CALL MULT(INGAMMA,ZRADIAL,SRAD,NCOEF)
          CALL MULT(CONST,SRAD,SPOLY,NCOEF)
          DO [illegible] K= 1,NCOEF
             ZTEMP(K)= NEWCOEF(K) [illegible] SPOLY(K)
3[illegible]          CPHASE2(I,J)= CPHASE2(I,J) + ZTEMP(K)
[illegible]       X= X + XINC
7[illegible]    CONTINUE
        Y= Y + YINC
3[illegible]  CONTINUE
C
C
C     AT THIS POINT THE COEFFICIENTS AND ESTIMATED W[illegible]-FRONTS
C     HAVE BEEN FOUND. [illegible] OF THE ALGORITHM [illegible]
C     [illegible] IS COMPUTED AND THE POLY[illegible]
C     CORR[illegible] ARE PRINTED OUT. WHEN [illegible] COEFFICIENTS
C     ARE BEING FOUND, THE CHARACTER ARRAY IS PRINTED OUT [illegible].
C
      CALL [illegible](PHEST,CPHASE1,ERROR1)
      CALL [illegible](PHASE,CPHASE2,ERROR2)
      IF(NCOEF .EQ. [illegible])CALL S[illegible](X[illegible],F[illegible],XC,YC,[illegible],[illegible])
C
      WRITE([illegible],[illegible])
33[illegible]  FORMAT("1")
      WRITE([illegible],8[illegible])
8[illegible]  FORMAT(" ",[illegible]X,"NEW COEFFICIENTS")
      IF(NCOEF .GT. 11)GO TO 6[illegible]
      IT=NCOEF
      GO TO [illegible]
3[illegible]  IT= 11
3[illegible]  WRITE([illegible],[illegible])(I,I=1,IT)
3[illegible]  FORMAT(" ",11("N(",I2,")",7X))
      WRITE([illegible],[illegible])(NEWCOEF(I),I=1,IT)
33[illegible]  FORMAT(" ",11(F[illegible],[illegible]X))
      IF(NCOEF .LE. 11) GO TO 6[illegible]
      WRITE([illegible],[illegible])(I,I=12,NCOEF)
```

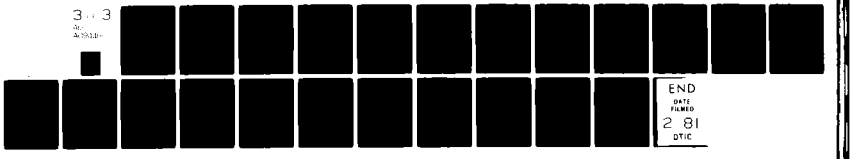
```
      WRITE (6,[illegible])(NEWCOEF(I),I=12,NCOEF)
33[illegible] WRITE (6,[illegible])
39[illegible] FORMAT(" ",[illegible]X,"ZERNIKE COEFFICIENTS")
      IF(NCOEF .GT. 11)GO TO [illegible]
      [illegible]
      [illegible]
3[illegible] [illegible]
3[illegible] WRITE(6,[illegible])(I,I=1,17)
3[illegible] FORMAT(" ",11("Z(",I2,")",7X))
      WRITE([illegible])(ZCOEF(I),I=1,17)
      IF(NCOEF .LE. [illegible])GO TO [illegible]
      WRITE(6,[illegible])(I,I=12,NCOEF)
      WRITE(6,[illegible])(ZCOEF(I),I=12,NCOEF)
33[illegible] WRITE(6,[illegible])[illegible]
3[illegible] FORMAT(" ",[illegible]X,"THE RMS ERROR USING THE ZERNIKE POLYS IS ",F[illegible])
      WRITE(6,[illegible])[illegible]2
33   FORMAT(" ",[illegible]X,"THE RMS ERROR USING THE NEW POLYS IS ",F[illegible])
33[illegible] CONTINUE
      STOP
      END
```

```
      SUBROUTINE GAMOUT(GY,NCOEF,BETA)
      DIMENSION GY(22,22)
C
C     THIS SUBROUTINE COMPUTES THE INNER PRODUCT COEFFICIENTS.
C     EACH OF THE G VARIABLES CORRESPOND TO [illegible]
C     ROUTINE RESULTS FROM THE INTEGRATION OF THE FOLLOWING
C     [illegible] THE INTEGRATION WAS DONE BY HAND TO
C     SPEED UP THE RUN TIME OF THIS ROUTINE AND TO IMPROVE ITS
C     ACCURACY.
C
C          M                        1.     M       (N-M-2)          M      M     2
C     GAMMA  = SQRT(2(M+1)*(INT     (R (P)  -  SUM  GAMMA  *  R (P))  DP)
C          N                        BETA   N       J=               NJ     J
C
C          M                        1.     M      M
C     GAMMA  = 2(M+J+1)*INT        R (R)*R  (R) * DR
C          NJ                      BETA   N      M+J
C
C
C     WHERE THE SUMMATION IS NOT PERFORMED IF N-M-2 < 1. [illegible]
C     [illegible] THE [illegible] POLYNOMIALS,
C     AND R (R)'S ARE THE ZERNIKE RADIAL POLYNOMIALS.
C
C     [illegible]
C     [illegible]
C     [illegible] VARIABLES [illegible]
C     [illegible]
C     [illegible]
C
C     BETA    : OBSCURATION RATIO
C     SUM  :    SIGMA SUPER N SUB ML
C     [illegible]  :  TEMPORARY VARIABLES
C     GY : ARRAY TO HOLD INNER PRODUCT COEFFICIENTS
C     NCOEF : NUMBER OF COEFFICIENTS BEING FOUND.
```

```
GT = 1.0 - BETA**2
G  = SQRT(GT)
GT1= 1. - BETA**4
G11= SQRT(GT1)
GT2= 1. - BETA**6
G22= SQRT(GT2)
GT3= 1. - BETA**8
G33= SQRT(GT3)
GT[illegible]= 1. - [illegible]**1[illegible]
G[illegible]= SQRT(GT[illegible])

GT[illegible]= 1.0 - BETA** 12
G[illegible]= SQRT(GT[illegible])
GT[illegible]= 1. - B[illegible]**1[illegible]
G[illegible]= SQRT(GT[illegible])
GT[illegible]= 1. - [illegible]**1[illegible]
G[illegible]= SQRT(GT[illegible])
GT[illegible]= 1. - BETA**16
G[illegible]= SQRT(GT[illegible])

G[illegible] = (BETA**2-BETA**4)/G[illegible]

TEMP1=1. +G[illegible]2 /G
AT=7.[illegible](GT[illegible])[illegible](TEMP1**2)
BT=5.[illegible](GT1)*TEMP1
CT=4.  (GT2)

G[illegible]22= SQRT(AT-BT+CT)

IF(N[illegible]EF .LE. [illegible])GO TO [illegible]
G[illegible]1= (2. /G111)*(BETA**[illegible]-BETA**[illegible]5)

TEMP2=2. +G131/G111
AT=2.[illegible](GT1)[illegible](TEMP2**2)
BT=3.[illegible](GT2)*TEMP2
CT=2.  (GT3)

G133= SQRT(AT-BT+CT)

G[illegible])= (1. /G[illegible])*(3.[illegible]*BETA**4 - BETA**2 - 2.[illegible]BETA**[illegible]5)

AT= 2.[illegible] (BETA *G) * (2.[illegible] + TEMP1)
```

```
[illegible]= ([illegible] [illegible]) [illegible] TEMP1
[illegible]= [illegible] ([illegible])
[illegible]= (BETA* [illegible]) [illegible] (1.  + 3./TEMP1)

[illegible]= (3. /G [illegible]) [illegible] (AT+BT-DT-CT)

[illegible] +2. [illegible] 2/G 22
[illegible] +(G [illegible]2/G [illegible]) TEMP1
[illegible]
[illegible] (GTD) TEMP2
[illegible] (GTD) [illegible] (TEMP [illegible] 2 + 12. [illegible]P )
[illegible] (G[illegible]) [illegible] TEMP
[illegible] ([illegible] 2)

[illegible]=[illegible] ([illegible]+DT-[illegible]T)

IF(NCOEF .LE. [illegible])GO TO [illegible]

G2 2=(7. /G222)*(ALTA**5 - BETA**5)

TEMP= [illegible]. + G2*2/G222
[illegible] ([illegible])
[illegible] (GTD) TEMP
[illegible] ([illegible])*(TEMP[illegible]2)

[illegible]=SQRT( [illegible]-[illegible]+CT)

[illegible]=(1/G111)*(3. [illegible](BETA**5)- [illegible] *([illegible])- [illegible] [illegible])

[illegible] ([illegible]) [illegible] TEMP2 + 1[illegible]. )
[illegible] ([illegible]) [illegible] TEMP
[illegible]
[illegible] ([illegible]) [illegible] (4.*TEMP2 + 3. )

[illegible]= (1. /G133) [illegible] (AT+BT-DT-CT)

TEMP[illegible]= 3. - ([illegible]/G111) + (G[illegible]/G133) [illegible]
TEMP[illegible]= [illegible]. + G[illegible]/G133
[illegible]
[illegible] TEMP
[illegible] (GTD) [illegible] (TEMP [illegible]2) + 2 . [illegible]))
[illegible]= 12. [illegible] GT2 + TEMP [illegible] TEMP1
[illegible] [illegible] GT1 [illegible] TEMP[illegible]**2
```

AD-A094 419 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 20/5
LASER WAVE-FRONT ANALYZER SOFTWARE IMPROVEMENT.(U)
DEC 80 R C SUDDUTH
UNCLASSIFIED AFIT/GEO/PH/80-10 NL



END
DATE FILMED
2 81
DTIC

```
G3*3= (4../G333) * (BETA**8 - BETA**10)

TEMP8= 4.0 + G353/G333
AT= 23.0 * GT3
BT= 12.0 * GT2 * TEMP8
CT= 1.5 * GT3 * TEMP8**2

G355= SQRT(AT-BT+CT)

G151= (1./G[illegible])*(1..8*BETA**6 +BETA**2 -5.0*BETA**[illegible] -6. *BETA**[illegible])

AT= [illegible] BETA **8 * (3.0 + TEMP1)
BT= BETA** * (1. + 6.*TEMP1)
CT= 2. * BETA** + (4. + 8.*TEMP1)
DT= [illegible] BETA **1
ET= BETA **2 * TEMP1

G152= (3./G122) * (AT+BT-CT-DT-ET)

AT= 29.0 * (GT1) * (9.. + TEMP3)
BT= 17.0 * (GT2) * (2.*TEMP3 + 5.*TEMP4 + 1.)
CT= [illegible] * (GT1) * TEMP4
DT= 1[illegible] GT[illegible]
ET= 2.0 * (GT3) * (16.*TEMP4 + 1[illegible].*TEMP4 + 3[illegible]. )
FT= 2.5 * (GT1) * (TEMP3 + 12.*TEMP4)

G154= (1./G144) * (AT+BT+CT-DT-ET-FT)

TEMP9= 5.. + G 5./G144
TEMP10= 12. - 2.*(G152/G122) + (G154/G144)*TEMP3
TEMP11= 1. +(G15./G1..)-(G152/G122)*TEMP1+(G /G )*TEMP.
AT= [illegible] (GT5)
BT= 28.. * (GT5) * TEMP9
CT= [illegible].6 * (GT4) * (1..*TEMP1 + 5.*(TEMP9**2))
DT= [illegible] * (GT3) *(1..5*TEMP11 + 3.*TEMP9*TEMP1 )
ET= (7./3.) * (GT2) *((12.*TEMP9*TEMP11) + TEMP1**2)
FT= 7.. * (GT1) * TEMP10 * TEMP11
GT= 7.. * (GT.) * TEMP11**2

G155= SQRT(AT-BT+CT-DT+ET-FT +GT)


51    GY(1, 1)= G [illegible]
      GY(2, 2)= G111
      GY(3, 3)= G111
```

```
      GY(4, 1)=  G[illegible]2[illegible]
      GY([illegible], [illegible])=  G[illegible]22
      GY([illegible], [illegible])=  G222
      GY([illegible], [illegible])=  G222
      IF(NCOEF .LE. [illegible]) GO TO 1[illegible]
      GY(7, 2)=  G131
      GY(7, [illegible])=  G133
      GY([illegible], [illegible])=  G131
      GY(3, [illegible])=  G133
      GY(3, [illegible])=  G333
      GY(1[illegible],1[illegible])=  G333
      GY(11,[illegible])=  G[illegible]
      GY(11,[illegible])=  G[illegible]2
      GY(11,11)=  G[illegible]
      IF(NCOEF .LE. 11) GO TO 1[illegible]
      GY(12,[illegible])=  G[illegible]2
      GY(1[illegible],12)=  G2[illegible]
      GY(13,[illegible])=  G2[illegible]2
      GY(1[illegible],13)=  G2[illegible]
      GY(1[illegible],1[illegible])=  G[illegible]
      GY(1[illegible],1[illegible])=  G[illegible]
      GY(1[illegible],2)=  G151
      GY(1[illegible],[illegible])=  G153
      GY(1[illegible],16)=  G1[illegible]
      GY(17,3)=  G1[illegible]1
      GY(1[illegible],[illegible])=  G1[illegible]
      GY(1[illegible],17)=  G1[illegible]
      GY(1[illegible],[illegible])=  G3[illegible]
      GY(1[illegible],18)=  G3[illegible]
      GY(1[illegible],1[illegible])=  G3[illegible]
      GY(1[illegible],1[illegible])=  G[illegible]
      GY(2[illegible],2[illegible])=  G[illegible]
      GY(21,21)=  G[illegible]
      GY(22,1)=  G[illegible]
      GY(22,[illegible])=  G[illegible]2
      GY(22,11)=  G[illegible]
      GY(22,22)=  G[illegible]
 1[illegible]   RETURN
      END
```

```
      SUBROUTINE FAPLR(XC,YC,FMIN,BETA)
      COMMON PHASE(32,32),SER(32,32),BAD
C
C
C     THIS ROUTINE FINDS THE CENTER, OUTSIDE
C RADIUS, AND THE OBSCURATION RATIO. THE ALGORITHM
C IS
C
C 1. USING THE DATA AT EACH CORNER, SELECT THE VALUE
C    WHICH IS THE SAME AT ANY THREE CORNERS AS THE VALUE
C    OF NO DATA, BAD.
C
C 2. IF THE POINT (16,16) IN PHASE CONTAINS A VALUE
C    OF BAD, GO TO STEP 3.
C
C 3. IF NOT, STARTING AT (16,16), SPIRAL OUTWARD UNTIL
C    A POINT WITH THE VALUE OF BAD IS FOUND OR THE RADIAL
C    DISTANCE FROM 16,16, IS GREATER THAN 7.5.
C
C 4. WHEN A BAD POINT IS FOUND, SET NEW CENTER TO COORD-
C    INATES OF THIS POINT. IF THE DISTANCE WAS GREATER
C    THAN 2.0, THEN KEEP THE CENTER AT (16,16).
C
C 5. BY CALLING EDGE, FIND THE VERTICAL AND HORIZONTAL
C    PIXEL OF THE INSIDE RADIUS OF THE ANNULUS. THE PIX-
C    ELVALS, AN ERROR CONDITION, THE WAVE-FRONT IS NON-
C    ANNULAR. THEREFORE, GO TO STEP 6. REPEAT UNTIL THE
C    RADII STABILIZE OR TEN ITERATIONS HAVE BEEN DONE.
C
C 6. FROM THE CENTER FOUND IN STEP 4, FIND THE SMALLEST
C    OUTSIDE RADIUS.
C
C 7. IF THE REGION WAS DEFINED AS NON-ANNULAR, DEFINE THE
C    WAVE-FRONT BY CALLING CONTUR AND FIND THE LARGEST
C    OUTSIDE RADIUS.  PASS THIS VALUE AND THE CENTER COORD-
C    INATES TO VALID TO FIND THE FLOATING-POINT VALUES OF
C    THE RADIUS AND CENTER.
C
C 8. IF THE REGION IS ANNULAR,USE CONTUR TO DEFINE THE REGION.
C    VALID IS CALLED TO FIND THE SMALLEST INSIDE RADIUS FOR AN
C    ANNULUS.  THE INSIDE RADIUS AND THE SMALLEST OUTSIDE
C    RADIUS IS USED TO DEFINE THE LARGEST POSSIBLE OUTSIDE RADIUS.
C
C 9. FILL THE CHARACTER ARRAY WITH FINAL SYMBOLS.
```

```
C     11. RETURN TO CALLING ROUTINE.
C
C     ***********************************************************************
C     ***********************************************************************
C     ************** DEFINITION OF VARIABLES ********************************
C     ***********************************************************************
C     ***********************************************************************
C
C     IS: CENTER OF THE COLUMNS
C     JS:   "     "   "  ROWS
C     IR: RADIUS WITH RESPECT TO THE COLUMNS
C     IR1:  "     "      "      "  "   ROWS
C     PHASE: CURRENT FRAME BEING PROCESSED
C     IOE: ODD/EVEN FLAG
C     IFLAG: USED IN THE CALL TO EDGE FOR DIRECTION AND ERROR
C     ICOUNT: MAX COUNT IN ONE DIRECTION FOR SPIRAL
C     ICN: COUNTER FOR TEN ITERATION OF EDGE.
C     FMIN: FINAL OUTSIDE RADIUS.
C     BAD: VALUE OF NO DATA.
C
C
      NR=32
      NC=32
      IRAD= [illegible]
C
C
C     THIS IS THE POINT WHERE THE VALUE OF
C     BAD IS SELECTED. THE IF'S SELECT THE
C     VALUE WHERE THREE MATCH.
C
      C1= PHASE(1,1)
      C2= PHASE(1,32)
      C3= PHASE(32,1)
      C4= PHASE(32,32)

      IF(C1 .EQ. C2 .AND. C1 .EQ. C3)BAD= C1
      IF(C1 .EQ. C2 .AND. C1 .EQ. C4)BAD= C1
      IF(C1 .EQ. C3 .AND. C1 .EQ. C4)BAD=C1
      IF(C3 .EQ. C2 .AND. C3 .EQ. C4)BAD=C3
      IS=16
      JS=16
      IR= [illegible]
```

```
      IR1= 
      ICN=1
      ITS=
      IFLAG=
      ICOUNT=1
      ITS=I
      JTS=JS
      IF (PHASE(IS,JS) .EQ. BAD) GO TO 1.
      GO TO 11
C
C     IF BAD WAS FOUND AT 16,16 THE ROUTINE CONTINUES ELSE IT GOES TO
C     11 TO START THE SPIRAL SEARCH.
C
 1    ICN= ICN + 1
      IFL=
      CALL EDGE(ITS,JTS,ILRAD,IRRAD,IFL)
C
C     IF EDGE RETURNS A 2 IN IFL THEN THE SPIRAL SEARCH HAD
C     HIT THE EDGE OF ENTIRE WAVE FRONT AND TRIED TO LOOK BEYOND
C     THE EDGE OF THE ARRAY. THUS THERE IS NO ANNULUS.
C     WITH THE RETURNED VALUES, THE NEW X AXIS ESTIMATED CENTER IS
C     COMPUTED. IF THERE IS NO CHANGE, THE NEXT CALL TO EDGE IS TRIED.
C
      IF(IFL .EQ. 2)GO TO 6000
      ID1= ILRAD + IRRAD
      IR1= ID1 / 2
      IF(ILRAD .EQ. IRRAD)GO TO 2
      IF(ILRAD .GT. IRRAD)GO TO 17
      ITS= ITS - ILRAD + IR1
      GO TO 2
 17   ITS= ITS + IRRAD - IR1
 2    IFL= 1
      CALL EDGE(ITS,JTS,IURAD,IDRAD,IFL)
C
C     AS ABOVE IF IFL IS 2 THEN THERE IS NO ANNULUS.
C     AGAIN THE RETURNED VALUES FROM EDGE ARE USED TO COMPUTE THE
C     Y AXIS CENTER. IF NO CHANGE THE ROUTINE WILL START THE ANALYSIS
C     FOR THE ACTUAL CENTER AND RADIUS.
C
      IF(IFL .EQ. 2)GO TO 6000
C
      ID= IURAD + IDRAD
      IR= ID / 2
      IF(IURAD .EQ. IDRAD)GO TO 3
      IF(IURAD .GT. IDRAD)GO TO 25
      JTS= JTS - IURAD + IR
      GO TO 3
```

```
250   ITS= ITS + IDRAD - IR
310   IF((IS .EQ. ITS .AND. JS .EQ. JTS) .OR. ICN .EQ. 1 )GO TO 
      ITS= ITS
      JS= JTS
      GO TO 1
C
C     AT THIS POINT EITHER THE CENTER HAS BEEN APPROXIMATED
C     OR ANOTHER ITERATION THROUGH EDGE IS REQUIRED. IF NOT,
C     THE ESTIMATED CENTER IS COMPUTED.
C
      FIS= FLOAT(IS)
      FJS= FLOAT(JS)
      IF(IS .NE. ITS)FIS= FLOAT(IS + ITS)/2.
      IF(JS .NE. JTS)FJS= FLOAT(JS + JTS)/2.
      IF(I+1 .GT. IR) GO TO 
      IRAD=IR
      GO TO 
      IRAD= IR1
      GO TO 
      TEMP=SQRT(FLOAT((IS-ITS)**2) + FLOAT((JS-JTS)** 2))
      IF(TEMP .GT. 5.75)GO TO 6
C
C
C     THIS IS THE START OF THE SPIRAL SEARCH. THE SPIRAL
C     WORKS BY ALL LEFT AND UP MOVES OCCUR WHEN ICOUNT IS ODD
C     AND RIGHT AND DOWN MOVES WHEN ICOUNT IS EVEN.
C     THE FIRST LINES OF CODE DETERMINE DIRECTION AND THE NEXT
C     STARTING AT 12 I, SECTION LOOKS UP.
C
11    IF(ICF .EQ. ) GO TO 12
      GO TO 
12    IF(IFLAG .EQ.  )GO TO 2
      I=1
13    IF(PHASE(IIS-I,JTS) .EQ. PHI)GO TO 14
      IF(I .EQ. ICOUNT)GO TO 15
      I=I+1
      GO TO 13
14    ITS=ITS-I
      GO TO 
15    IFLAG=
      ICF=1
      ITS=ITS-I
      ICOUNT=ICOUNT+1
      GO TO 1
C
C     THIS SECTION SEARCHS RIGHT IN PHASE FROM PHASE(ITS,JTS)
```

```
C      TO PHASE(ITS,JTS+ICOUNT)
C
2100   I=1
2110   IF(PHASE(ITS,JTS+I) .EQ. BAD)GO TO 220
       IF(I .EQ. ICOUNT)GO TO 230
       I=I+1
       GO TO 21
2200   JTS=JTS+1
       GO TO 5
2300   IFLAG=1
       JTS=JTS+I
       GO TO 11
C
C      THIS SECTION SEARCHS DOWN IN PHASE FROM PHASE(ITS,JTS)
C      TO PHASE(ITS,JTS+ICOUNT)
C
3100   IF(IFLAG .EQ.  )GO TO 4
       I=1
3110   IF(PHASE(ITS+1,JTS) .EQ. BAD)GO TO 32
       IF(I .EQ. ICOUNT)GO TO 33
       I=I+1
       GO TO 31
3200   ITS=ITS+I
       GO TO 5
3300   I=
       IFLAG=
       ICOUNT=ICOUNT+1
       ITS=ITS+I
       GO TO 1
C
C      THIS SECTION SEARCHS LEFT IN PHASE FROM PHASE(ITS,JTS)
C      TO PHASE(ITS,JTS-ICOUNT)
C
4100   I=1
4110   IF(PHASE(ITS,JTS-I) .EQ. BAD) GO TO 42
       IF(I .EQ. ICOUNT)GO TO 43
       I=I+1
       GO TO 41
4200   JTS=JTS-1
       GO TO 5
4300   IFLAG=1
       JTS=JTS-I
       GO TO 1
5000   IS=ITS
```

```
      JS=JTS
      GO TO 10
C
C
C     SEARCH FOR SMALLEST OUTSIDE RADIUS
C
 600  IOUT=IR+1
 610  IF(PHASE(IS,JS+IOUT) .EQ. BAD)GO TO 620
      IF(JS+IOUT .EQ. 32)GO TO 620
      IOUT=IOUT+1
      GO TO 610
 620  IOUT1=IR + 1
 630  IF(PHASE(IS,JS-IOUT1) .EQ. BAD)GO TO 640
      IF(JS-IOUT1 .EQ. 1)GO TO 640
      IOUT1=IOUT1+1
      GO TO 630
 640  IDOUT=IR+1
 650  IF(PHASE(IS+IDOUT,JS) .EQ. BAD)GO TO 660
      IF(IS+IDOUT .EQ. 32)GO TO 660
      IDOUT = IDOUT + 1
      GO TO 650
 660  IUOUT=IR+1
 670  IF(PHASE(IS-IUOUT,JS) .EQ. BAD)GO TO 680
      IF(IS-IUOUT .EQ. 1)GO TO 680
      IUOUT=IUOUT + 1
      GO TO 670
 680  MINLR=IOUT1
      IF(IOUT .LT. IOUT1)MINLR= IOUT
      MINUD=IDOUT
      IF(IUOUT .LT. IDOUT)MINUD= IUOUT
      MIN= MINUD
      IF(MINLR .LT. MINUD) MIN= MINLR
C
C
C     IF THE REGION IS NON-ANNULAR, THEN TO GO 7..0
C     ELSE CALL VALID TO FIND THE CENTER AND THE INSIDE
C     RADIUS. ONCE FOUND, FILL THE CENTRAL REGION WITH #
C     THEN COMPUTE THE OBSCUTATION RATIO AND RETURN.
C     BEFORE FILLING THE ARRAY, THE LARGEST
C     OUTSIDE RADIUS IS FOUND BY SEARCHING OUTSIDE
C     THE INNER RADIUS UNTIL THE OUTSIDE RADIUS IS
C     JUST BIG ENOUGH TO NOT FIND ANY BAD POINTS IN
C     ITS REGION OF SEARCH.
      IF(IRAD .EQ. 0)GO TO 7000
```

```
      CALL CONTUR(15.,15.,-1.)
      CALL VALID(IS,JS,IRAD,XC,YC,RADIUS,1)
      FLG=
      FMIN= FLOAT(MIN)
5440  DO 55 I= 1,32
          DO 55   J= 1,32
               RHO= SQRT((FLOAT(I)-YC)**2 + (FLOAT(J)-XC)**2)
               IF(RHO .GE. FMIN .OR. SER(I,J) .EQ. 1H#)GO TO 55
               IF(PHASE(I,J) .EQ. BAD)FLG= 1
55 0      CONTINUE
55 0  CONTINUE
      IF(FLG .EQ. 0)GO TO 6700
      FMIN= FMIN - .2*
      FLG=
      GO TO 641
57 0  BETA= RADIUS / FMIN
      GO TO 9999


      FOR A NON-ANNULAR REGION, FIND THE LARGEST RADIUS
      THEN CALL VALID TO FIND THE TRUE CENTER AND RETURN
      THE CENTER AND OUTSIDE RADIUS TO THE CALLING ROUTINE.

70 0  CALL CONTUR(15.,15.,-1.)
      IRAD= IOUT1
      IF(IOUT .GT. IOUT1)IRAD= IOUT
      IF(IOUT .GT. IRAD)IRAD= IOUT
      IF(IOUT .GT. IRAD)IRAD= IOUT
      CALL VALID(IS,JS,IRAD,XC,YC,RADIUS,2)
      FMIN= RADIUS
      BETA=  .

9999  DO 7110 I= 1,32
          DO 7100   J= 1,32
               RHO=SQRT((FLOAT(I)-YC)**2 + (FLOAT(J)-XC)**2)
               IF(RHO .GT. FMIN .OR. SER(I,J) .EQ. 1H#)GO TO 7100
               SER(I,J)= 1H^
7100      CONTINUE
71 0  CONTINUE
      RETURN
      END
```

```
      SUBROUTINE VALID(IS,JS,IRAD,XC,YC,RAD,IN)
      COMMON PHASE(32,32), SER(32,32),BAD
C
C     THIS ROUTINE FINDS THE CENTER OF THE UNOBSCURED PORTION
C  OF THE BEAM.  IT DOES THIS BY FINDING THE FIRST MOMENTS
C  OF THE I'S AND J'S.  BEFORE FIGURING THE MOMENTS, IT
C  FIRST (FLAG=1) USES THE APPROXIMATE CENTER (IS,JS) AND
C  THE RADIUS PLUS 2 (IRAD) TO FIND THE NUMBER OF GOOD POINTS
C  IN THE CENTER OF THE WAVE-FRONT.  AT THE SAME TIME IT
C  FINDS THE SUM OF THE I'S AND J'S FOR THE Y AND X MOMENTS
C  RESPECTIVELY (M1 & M1 ).  THE CENTER IS FOUND BY
C  X = M1/#POINTS AND Y = M1/#POINTS.  WITH X AND Y
C  THE PROGRAM THEN TRIES TO FIND THE SMALLEST RADIUS BY USING
C  IRAD AS AN INITIAL GUESS.  THE RADIUS IS INCREMENTED BY
C  .25 UNTIL ALL OF THE POINTS FOUND ABOVE ARE FOUND.  THUS,
C  THE SMALLEST CIRCLE WHICH WILL CIRCUMSCRIBE THE GOOD PHASE
C  IS FOUND.  WHEN IN IS 1 THE PHASE IS AN ANNULUS.  WHEN IT IS
C  THIS ROUTINE WILL LOOK FOR THE LARGEST CIRCLE TO FILL
C  ALL OF THE GOOD PHASE.
C
      FLAG=0
      PNTS=0
      NPTS=0
      M1 =0
      M11=0
      RAD=FLOAT(IRAD)
C
C     IF THERE IS NO ANNULUS (IN=0) THEN SEARCH THE ENTIRE REGION
C     FOR THE CENTER OF THE VALID DATA.
C
      IF(IN .EQ. 0)GO TO 50
      ITS= IS - IRAD - 1
      ITTS= IS + IRAD + 2
      JTS= JS - IRAD - 1
      JTTS= JS + IRAD + 2
      IF(ITS .LE. 0 .OR. JTS .LE. 0)GO TO 13.
      IF(JTTS .GT. 32 .OR. ITTS .GT. 32)GO TO 13.
      GO TO 10
C
   51 ITS= 1
      JTS= 1
      ITTS= 32
```

```
      JTTS= 32

C
C     THIS SECTION DOES BOTH THE SEARCH FOR THE CENTER AND
C     SEARCH FOR THE BEST RADIUS. WHEN FLAG IS .1T COMPUTES
C     THE CENTER VALUES. WHEN ONE IT LOOKS FOR THE BEST RADIUS.
C
100   DO 700 I= ITS,ITTS
         DO 600 J= JTS,JTTS
            IF(FLAG .EQ. 1)GO TO 400
            IF(I .EQ. 1)GO TO 200
            IF(PHASE(I,J) .EQ. BAD)GO TO 600
            GO TO 300
200         IF(PHASE(I,J) .NE. BAD)GO TO 600
300         M1= M1 + J
            M2= M2 + I
            PNTS= PNTS + 1
            GO TO 600
400         RHO=SQRT((FLOAT(I)-Y0)**2 +(FLOAT(J)-X0)**2)
            IF(I .EQ.1)GO TO 500
            IF(RHO .GT. RAD)GO TO 600
            IF(PHASE(I,J) .EQ. BAD)NPTS= NPTS+1
            GO TO 600
500         IF(RHO .GT. RAD)GO TO 600
            IF(PHASE(1,J) .EQ. BAD)NPTS= NPTS+1
600      CONTINUE
700   CONTINUE

C
C     COMPUTE THE CENTER VALUES
C
      IF(FLAG .EQ. 1)GO TO 800
      X0= FLOAT(M1)/PNTS
      Y0= FLOAT(M2)/PNTS
      FLAG=1
      GO TO 100

C
C     COMPUTE THE SMALLEST INSIDE RADIUS.
C
800   IF(I .EQ. 1)GO TO 850
      IF(NPTS .EQ. PNTS)GO TO 900
      RAD = RAD + 0.25
      NPTS=0
      GO TO 100
```

```
C
C     COMPUTE THE LARGEST OUTSIDE RADIUS.
C
 850  IF(NPTS .EQ. [illegible])GO TO 900
      RAD= RAD - [illegible].2
      NPTS= [illegible]
      GO TO 40[illegible]
C
C     THIS IS DONE ONLY IF THERE IS AN INSIDE RADIUS. THUS IF
C     IN IS ZERO THE ROUTE STOPS.
 900  IF(IN .EQ. [illegible])GO TO 13[illegible]
      DO 120 I= IIS,IITS
         DO 110 J= JIS,JITS
            RHO=SQRT((FLOAT(I)-Y)**2 + (FLOAT(J)-X)**2)
 100        IF(RHO .LE. RAD)SER(I,J)= 1H*
 110     CONTINUE
 120  CONTINUE
      RETURN
 130  WRITE(6,14[illegible])
 140  FORMAT(" ",IX,"VALID IS TRYING TO SEARCH OUTSIDE OF PHASE")
 150  RETURN
      END
```

```
      SUBROUTINE EDGE(IUD,ILR,IRAD,IRAD1,IFLAG)
C
C     EDGE FINDS THE HORIZONTAL OR VERTICAL RADII, DEPENDING
C     ON THE VALUE OF IFLAG. WHEN IFLAG IS   1  SEARCHES
C     VERTICALLY AND HORIZONTALLY WHEN 1. USING IUD,ILR
C     AS THE STARTING POINTS, EDGE SEARCHES DOWN OR RIGHT
C     UNTIL AN EDGE IS FOUND. THEN THE
C     OPPOSITE DIRECTION. IRAD HOLDS THE LEFT/UP
C     RADIUS, AND IRAD1 THE RIGHT/DOWN. IT IS A DIRECTION
C     FLAG. IFLAG IS SET TO 2 IF IT TRIES TO GO OUT TO BOUNDS.
      COMMON PHASE(32,32),SAV(32,32),BAD
      II=1
      I=1
  100 IF(IFLAG .EQ. 2)GO TO 2
      IF(PHASE(IUD+1,ILR) .EQ. BAD)GO TO 3
      GO TO 300
  200 IF(PHASE(IUD,ILR+I) .EQ. BAD)GO TO 3
  300 IF(II .EQ. 1) GO TO 4
      IRAD=-I
      GO TO 5
  400 IRAD1=I
      II=-1
      I=-1
      GO TO 100
  500 I=I+I
C     THIS SECTION WILL LOOK FOR OUT OF BOUND ERRORS
      IF(IUD+I .GT. 32 .OR. ILR+I .GT. 32)GO TO 6
      IF(IUD+I .LT. 1  .OR. ILR+I .LT. 1 )GO TO 6

      GO TO 100
  600 RETURN
  700 IFLAG=2
      RETURN
      END
```

```
      SUBROUTINE ZRAD(ARY,ISIZE,X,Y)
C
C     THIS SUBROUTINE WILL FIND THE RADIAL POLYNOMIAL OF THE ZERNIKE TYPE
C     THE ARRAY ARY IS A COLUMN MATRIX AND IS FILLED TO
C     THE NUMBER OF COEFFICIENTS REQUESTED (NCOEF).
C
      DIMENSION ARY(22)
      ARY(1)= 1.
      ARY(2)= SQRT(X**2 + Y**2)
      ARY(3)= ARY(2)
      ARY(4)= 2.*(X**2 + Y**2) - 1.
      ARY(5)= X**2 + Y**2
      ARY(6)= ARY(5)
      IF(ISIZE .LE. 6)RETURN
      ARY(7)= ARY(2) * (3.*ARY(5) -2. )
      ARY(8)= ARY(5)
      ARY(9)= ARY(2) * ARY(8)
      ARY(10)= ARY(9)
      ARY(11)= 6.*ARY(8)**2 - 6.*ARY(8) + 1.
      IF(ISIZE .LE. 11) RETURN
      ARY(12)= ARY(8) * (4.*ARY(8) - 3.)
      ARY(13)= ARY(12)
      ARY(14)= ARY(8)**2
      ARY(15)= ARY(14)
      ARY(16)= ARY(8) * (10.*ARY(14) - 12.*ARY(8) +3.)
      ARY(17)= ARY(16)
      ARY(18)= ARY(8) * (5.*ARY(8) - 4. )
      ARY(19)= ARY(18)
      ARY(20)= ARY(8)*ARY(8)
      ARY(21)= ARY(20)
      ARY(22)= 2.*ARY(14)*ARY(8) - 3.*ARY(14) + 12.*ARY(8) - 4.

      RETURN
      END
```

```
      SUBROUTINE ZANG(ARRAY,ISIZE,X,Y)
C
C     THIS SUBROUTINE WILL SOLVE FOR THE ANGULAR AND AMPLITUDE
C FOR THE ZERNIKE POLY'S.
C THE ROUTINE FIRST COMPUTE THE SORT TERMS AND THE
C SINE AND COSINE TERMS.  THEN THE SQUARE DIAGONAL
C MATRIX IS FILLED TO THE PROPER AMOUNT AS DETERMINED BY
C ISIZE.
C
      DIMENSION ARRAY(22,22)
C
      A = X**2 + Y**2
      B = SQRT(A)
      C = SQRT(3.)
      D = SQRT(6.)
      E = SQRT(8.)
      F = SQRT(12.)
C
      COS1= X/B
      SIN1= Y/B
      COS2= (X**2 - Y**2)/A
      SIN2= (2.*X*Y)/A
      COS3= (X/B) * ((4.*A*X**2)/A - 3.)
      SIN3= (Y/B) * (3. - (4.*A*Y**2)/A)
      COS4= ((6.*X**4)/A**2) - (6.*X**2)/A + 1.
      SIN4= ((4.*X*Y)/A) * (4. - (2.*A*Y**2)/B)
      COS5= (X/B) * ((16.*X**4)/B**2 - (20.*X**2)/B + 5.)
      SIN5= (Y/B) * (5. - (20.*Y**2)/B + (16.*Y**4)/A**2)
C
      ARRAY(1,1)= 1.
      ARRAY(2,2)= COS1
      ARRAY(3,3)= SIN1
      ARRAY(4,4)= SQRT(3.)*A
      ARRAY(5,5)= C * SIN2
      ARRAY(6,6)= C * COS2
      IF(ISIZE .LE. 6) RETURN
      ARRAY(7,7)= D * SIN3
      ARRAY(8,8)= D * COS3
      ARRAY(9,9)= D * SIN3
      ARRAY(10,10)= D * COS3
      ARRAY(11,11)= SQRT(8.)*A
      IF(ISIZE .LE. 11) RETURN
      ARRAY(12,12)= E * COS2
```

```
      ARRAY(13,13)= E   *  SIN2
      ARRAY(14,14)= E   *  COS4
      ARRAY(15,15)= E   *  SIN4
      ARRAY(16,16)= E   *  COS
      ARRAY(17,17)= E   *  SIN
      ARRAY(18,18)= E   *  COS3
      ARRAY(19,19)= E   *  SIN3
      ARRAY(20,20)= E   *  COS5
      ARRAY(21,21)= E   *  SIN5
      ARRAY(22,22)= SQRT(2.0)

      RETURN
      END
```

198

```
      SUBROUTINE CONTUR(XC,YC,RAD)
      COMMON PHASE(32,32),SER(32,32),BAD
C
C     CONTUR FILLS THE ARRAY SER WITH THE PROPER SYMBOLS
C     IS USED THE INSIDE RADIUS AND THE CENTER, TO DECIDE
C     ON WHAT TO PUT IN THE ARRAY.
C
      DO 1 I=1,32
      DO 1 J=1,32
            SER(I,J)= 1H.
            IF(PHASE(I,J) .EQ. BAD)SER(I,J)= 1H*
            RHO= SQRT((J - XC)**2 + (I - YC) **2)
            IF(RHO .LE. RAD)SER(I,J)= 1H*
1     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE RMSERR(TRUE,GUESS,ERROR)
C
C     THIS SUBROUTINE WILL FIND THE RMS ERROR BETWEEN THE TRUE
C  PHASE-FRONT AND THE RECONSTRUCTED PHASE-FRONT.
C  THE PHASE IS NORMALIZED BY THE NUMBER OF 'GOOD PHASE POINTS'
C  IN THE ARRAY TRUE.
C
      DIMENSION TRUE(32,32),GUESS(32,32)
      COMMON PHASE(32,32),SER(32,20),BAD
C
      NPTS= 0
      SUM= 0.0
      DO 20 I= 1,32
         DO 10 J= 1,32
            IF(TRUE(I,J) .NE. 1H^)GO TO 10
            NPTS= NPTS + 1
            SUM= SUM + (TRUE(I,J) - GUESS(I,J)) **2
   10    CONTINUE
   20 CONTINUE
      ERROR= SQRT(SUM/FLOAT(NPTS))
      RETURN
      END
```

```
      SUBROUTINE SETPRNT(XRN,FRM,XC,YC,RADIUS,BETA,IUF)
C
C     THIS SUBROUTINE WILL PRINT OUT THE ARRAY OF WHICH
C     HOLDS INFORMATION ON WHERE THE MAIN PROGRAM WILL DO
C     THE PROCESSING. IT WILL ALSO PRINT THE PARAMETERS WHICH
C     WILL BE USED BY THE MAIN PROGRAM.
C
      COMMON PHASE(32,32),SET(32,32),[illegible]
C
      IF(IUF .EQ. 1)WRITE(6,1[illegible])XRN,FRM
      IF(IUF .[illegible]. 1)WRITE(6,1[illegible])XRN,FRM
 110  FORMAT("1",1[illegible]X,"FINAL RESULTS FOR RUN",1[illegible]2.[illegible],"  ,FRAME ",[illegible])
 15[illegible]  FORMAT("1",1[illegible]X,"INITIAL PARAMETERS FOR RUN",1[illegible]2.[illegible],"  ,FRAME ",[illegible])
      WRITE(6,17[illegible])
 175  FORMAT(" ",23X,"X AXIS")
      WRITE(6,2[illegible])
 2[illegible]  FORMAT(" ",23X,1 ("1",1X),1[illegible]("2",1X),3("3",1X))
      WRITE(6,3[illegible])
 3[illegible]  FORMAT(" ",[illegible]X,3(" 1 2 3 [illegible]"),"  1 2")
      DO [illegible] I=1,32
      IF(I .EQ. 2)WRITE(6,[illegible])I,(SER(I,J),J=1,32)
      IF(I .EQ. [illegible])WRITE(6,[illegible])I,(SER(I,J),J=1,32)
      IF(I .EQ. [illegible])WRITE(6,[illegible])I,(SER(I,J),J=1,32)
      IF(I .EQ. [illegible])WRITE(6,[illegible])I,(SER(I,J),J=1,32)
      IF(I .EQ. 1[illegible])WRITE(6,[illegible])I,(SER(I,J),J=1,32),XC,YC
      IF(I .EQ. 12)WRITE(6,[illegible])I,(SER(I,J),J=1,32),RADIUS
      IF(I .EQ. [illegible])WRITE(6,[illegible])I,(SER(I,J),J=1,32),BETA
      IF(I .EQ. [illegible])WRITE(6,12[illegible])I,(SER(I,J),J=1,32)
      IF(I .EQ. [illegible])WRITE(6,1[illegible])I,(SER(I,J),J=1,32)
      IF(I .EQ. [illegible])WRITE(6,[illegible])I,(SER(I,J),J=1,32)
      IF(I .EQ. 21)WRITE(6,1[illegible])I,(SER(I,J),J=1,32)
      IF(I .EQ. 22)WRITE(6,1[illegible])I,(SER(I,J),J=1,32)
      IF(I .EQ. 2 .OR. I .EQ. 4 .OR. I .EQ. [illegible] .OR. I .EQ. [illegible] .OR.
     * I .EQ. 1[illegible] .OR. I .EQ. 12 .OR. I .EQ. 1[illegible] .OR. I .EQ. [illegible] .OR.
     * I .EQ. 1[illegible] .OR. I .EQ. 2[illegible] .OR. I .EQ. 21 .OR. I .EQ. 22)
     * GO TO 4[illegible]
      WRITE(6,11[illegible])I,(SER(I,J),J=1,32)
 4[illegible]  CONTINUE
 45[illegible]  FORMAT(" ",2X,I2,32(1X,A1),3X,"  = NO DATA PRESENT.")
 [illegible]  FORMAT(" ",2X,I2,32(1X,A1),3X,". = VALID DATA.")
 [illegible]  FORMAT(" ",2X,I2,32(1X,A1),3X,"# = CENTRAL REGION OF ANNULUS.")
 [illegible]  FORMAT(" ",2X,I2,32(1X,A1),3X,"^ = DATA USED BY PROGRAM.")
```

```
80    FORMAT(" ",2X,I2,32(1X,A1),5X,"CENTER AT(",F5.2," , ",F5.2,")")
90    FORMAT(" ",2X,I2,32(1X,A1),5X,"RADIUS IS ",F5.2)
100   FORMAT(" ",2X,I2,32(1X,A1),5X,"OBSCURATION RATIO IS ",F5.4)
110   FORMAT(" ",2X,I2,32(1X,A1))
120   FORMAT(" ","Y",1X,I2,32(1X,A1))
130   FORMAT(" ","A",1X,I2,32(1X,A1))
140   FORMAT(" ","Y",1X,I2,32(1X,A1))
150   FORMAT(" ","I",1X,I2,32(1X,A1))
160   FORMAT(" ","S",1X,I2,32(1X,A1))
      RETURN
      END


      SUBROUTINE INVERT(A,B,MAX)
C     THIS SUBROUTINE WILL FIND THE INVERSE OF A SPARSE,
C      LOWER TRIANGULAR MATRIX

      DIMENSION A(22,22),B(22,22)

      DO 3  I=1,MAX
         DO 2  J=1,I
               IF(A(I,J) .EQ. 0.0 )GO TO 2
               B(I,J)= -(A(I,J)/A(I,I))
               IF(I .EQ. J)B(I,J)= 1./A(I,J)
2        CONTINUE
3     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE MULT(A,B,C,L)
C     THIS IS A SIMPLE MATRIX MULTIPLY ROUTINE
C     IT MULTIPLIES A L BY L ARRAY TIMES A L ARRAY , B
C     AND PUTS THE RESULTS INTO ARRAY C.
      DIMENSION A(22,22),B(22),C(22)
         DO 20 I=1,L
            C(I)=0.
            DO 10 J=1,L
               C(I)= C(I) + A(I,J) * B(J)
10          CONTINUE
20       CONTINUE
      RETURN
      END

      SUBROUTINE ARRP1(A,ISIZE,NAME)
C     THIS ROUTINE PRINTS OUT AN COLUMN OR ROW
C      MATRIX. IT ALSO HAS A HOLLERITH STRING PASSED
C      IN NAME. IT IS USED ONLY FOR DEBUG PURPOSES.
      DIMENSION A(22)
      WRITE(6,5)NAME
5     FORMAT(" ",2X,A10)
      WRITE(6,10)(A(I),I=1,ISIZE)
10    FORMAT(" ",11(2X,F10.5))
      RETURN
      END

      SUBROUTINE ARRP2(A,ISIZE,NAME)
C     THIS ROUTINE SIMPLY PRINTS A ISIZE BY ISIZE
C      ARRAY. A HOLLERITH STRING IS PASSED IN NAME
C      TO LABEL THE ARRAY BEING PRINTED. THIS ROUTINE
C      IS USED FOR DEBUGGING.
      DIMENSION A(22,22)
      WRITE(6,5)NAME
5     FORMAT(" ",2X,A10)
      DO 20 I=1,ISIZE
         WRITE(6,10)(A(I,J),J=1,ISIZE)
10    FORMAT(" ",11(2X,F10.5))
20    RETURN
      END
```

VITA

Lt. Robert C. Sudduth was born on April 30, 1954 in Toledo, Ohio. He graduated from Brookfield Central High School in Brookfield, Wisconsin in 1972. He enlisted in the United States Air Force that fall and became a Weapons Control Systems Mechanic on F-106A/B's at Tyndall AFB, Florida. In August 1976 Lt. Sudduth received an AFROTC scholarship to attend Purdue University, from which he graduated with the degree of Bachelor of Science in Electrical Engineering in December 1978. At the same time he received his commission, and was a Distinguished AFROTC Graduate. In June 1979, Lt. Sudduth entered AFIT. He is a member of Eta Kappa Nu and Tau Beta Pi.

Permanent address: 1225 Indianwood Dr.
Brookfield, WI 53005

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: AFIT/GEO/PH/80-10	2. GOVT ACCESSION NO.: AD-A094419
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): LASER WAVE-FRONT ANALYZER SOFTWARE IMPROVEMENT	5. TYPE OF REPORT & PERIOD COVERED: MS Thesis
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Robert C. Sudduth, 2dLt., USAF	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Air Force Institute of Technology(AFIT-EN) Wright-Patterson AFB, Ohio 45433	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: AFWL Kirtland AFB, New Mexico	12. REPORT DATE: December 1980
	13. NUMBER OF PAGES: 213
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES
Approved for public release; IAW AFR 190-17

06 JAN 1981

Frederick C. Lynch, Major, USAF
Director of Public Affairs

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Zernike Polynomials
Gram-Schmidt Orthogonalization
Wave-front Analysis

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
A software package was written which will analyze annular shaped laser wave-fronts. The polynomials used to estimate the wave-front are based on the work of J.Y. Wang and D.E. Silva in their paper "Wave-front Interpretation with Zernike Polynomials." This involved generating a set of orthogonal polynomials from the Zernike polynomials by using the Gram-Schmidt orthogonalization process. The coefficients of the polynomials are determined by

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

using the orthogonality of the polynomials, instead of using the common least-squares method. The coefficients of the generated polynomials are converted to Zernike coefficients, and both sets of coefficients are presented to the user.

By using the first moments of the wave-front's position in the collection array, the software is able to define the basic parameters of the wave-front. These parameters are: center, outside radius, and the obscuration ratio of the wave-front. With these parameters, the soft-ware computes 6, then 11, then 22 coefficients to show the stability of the coefficients. With well-defined circular and annular wave-fronts, the program was consistantly able to compute the coefficients with an RMS error of less than 0.050 waves.